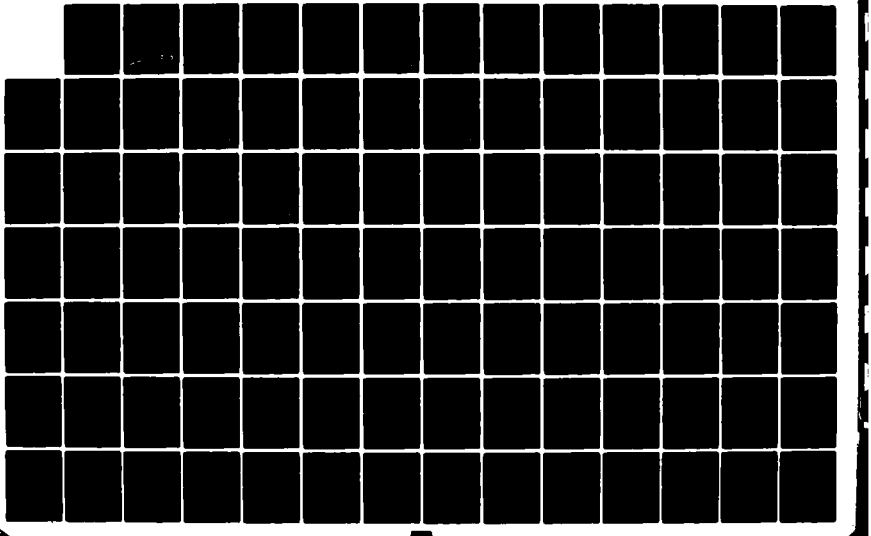
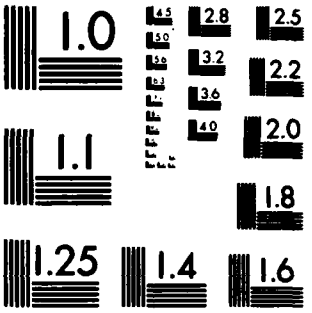
AD A124 998 ADA* SOFTWARE DESIGN METHODS FORMULATION(U) SOFTECH INC 1/3
WALTHAM MA OCT 82 DAAK80-80-C-0187

UNCLASSIFIED F/G 9/2 NL





(11)

ADA* SOFTWARE DESIGN METHODS FORMULATION

FINAL REPORT

OCTOBER 1982

CENTER FOR TACTICAL COMPUTER SYSTEMS
(CENTACS)

U. S. ARMY COMMUNICATIONS - ELECTRONICS COMMAND
(CECOM)

CONTRACT DAAK80-80-C-0187

PREPARED BY

SOFTECH, INC.
460 TOTTEN POND ROAD
WALTHAM, MA 02154

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1 REPORT NUMBER	2 GOVT ACCESSION NO. AD-A124 998 / 3 RECIPIENT'S CATALOG NUMBER
4 TITLE (and Subtitle) Ada Software Design Methods Formulation: Final Report	5 TYPE OF REPORT & PERIOD COVERED
	6 PERFORMING ORG. REPORT NUMBER
7 AUTHOR(s)	8 CONTRACT OR GRANT NUMBER(s) DAAK80 - 80 - C - 0187
9 PERFORMING ORGANIZATION NAME AND ADDRESS SofTech, Inc. 460 Totten Pond Road Waltham, MA. 02154	10 PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11 CONTROLLING OFFICE NAME AND ADDRESS United States Army Communications Electronics Command Fort Monmouth, N.J. 07703	12 REPORT DATE October 1982
	13 NUMBER OF PAGES
14 MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15 SECURITY CLASS. (of this report) Unclassified
	15a DECLASSIFICATION/DOWNGRADING SCHEDULE

16 DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17 DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18 SUPPLEMENTARY NOTES

This report is one of three in a series entitled; "ADA SOFTWARE DESIGN METHODS FORMULATION". Other Reports include; "Case Studies." and "Appendices to Final Report".

19 KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer programming, software, design and development of computers and peripheral equipment, programming languages, professional development, curriculum development, training and education, on job training.

20 ABSTRACT (Continue on reverse side if necessary and identify by block number)

The "Final Report" recommends an overall approach to Ada training and suggests a baseline from which a formal Ada education program can follow. It includes; establishing a generic job classification schema for the embedded systems community, identification of Ada knowledge requirements for each job category and recommendations for an Ada training curriculum. The report also suggests appropriate training methods for embedded systems programmers. The reader is referred to the "Case Studies Report" for a discussion of Ada's impact on the software life cycle.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

Section 1

ABSTRACT

The Ada Software Design Methods Formulation contract was performed by SofTech for the U.S. Army Communication-Electronics Command (CECOM) in order to 1) formulate and document effective software design methods in Ada, and 2) to establish a training baseline from which a formal Ada education program can follow. The first of these goals required SofTech to observe the efforts of two contractors funded to redesign existing large military systems in Ada and to extract case studies illustrating significant issues which arose as a result of the introduction of Ada into the software life cycle. (These studies appear in the Case Studies Report, submitted as a separate item under this contract.) The second involved establishing a generic job classification schema for the embedded systems community, identifying Ada knowledge requirements for each job category, recommending an Ada training curriculum based on these requirements, and determining appropriate training methods for embedded systems programmers.

The present report addresses the results of the latter effort. (The reader is referred to the Case Studies Report for a discussion of Ada's impact on the software life cycle.) Stated briefly, SofTech's findings are as follows:

1. Based upon a functional decomposition of the embedded systems work force, the following generic job categories have emerged:

   PROJECT ADMINISTRATIVE MANAGER
   SENIOR ENGINEERING MANAGER
   SUPPORT MANAGER
   PROJECT/TASK LEADER
   CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
   DESIGN CONSULTANT
   PROGRAMMER
   SOFTWARE DESIGNER
   REAL-TIME SYSTEM ARCHITECT
   SPECIALIST
   JUNIOR STAFF MEMBER/TECHNICAL AIDE

SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
SYSTEM INTEGRATION ENGINEER

These categories are covered in detail in Sections 3 and 4.

2. Given the Ada knowledge requirements of each category, SofTech has recommended a three-pronged curriculum, incorporating training in the Ada Programming Support Environment (see Figure 1-1), the Ada language (see Figure 1-2), and software engineering methodologies (see Figure 1-3). The suggested progression through the curriculum is shown in Figure 1-4. This course sequence will vary by job category (see curriculum paths in Section 5.3) and by individual experience.

3. The embedded systems community is very receptive to training, with adequate classroom and video tape facilities, and with a definite bias toward a lecture/workshop format for training programmers.

NO.	TITLE	DESCRIPTION	DURATION
E101	APSE Concepts for Technical Managers	broad overview of APSE emphasizing how it supports software life cycle	1 day
E102	APSE Overview for Programmers	broad overview of APSE for software developers	1/2 day
E103	Basic APSE Operation	introduction to APSE concepts, basic editing, etc., for people who will not be real users	1/2 day
E201	User's Introduction to the APSE	basic use of the APSE database, file system, command language; tool overview	3 days
E301	Command Language	command language, substitutors, I/O redirections	1 day
E302	Program Development	compiler, linker, exporter, loader	2 days
E303	Database	files, directories, attributes, associations, access control, node sharing, program libraries, etc.	2 days
E304	Debugging	debugger, timing analyzer, frequency analyzer	1 1/2 days
E305	Assembling and Importing	assembly language, importer	1/2 day
E306	Configuration Management and Program Management	tools to support CM and PM, example tools one might build	3 days
E401	How to Add Tools	programming with the command language, KAPSE tool interfaces, examples of useful tools	2 days
E402	System Administrator's Course	user authorization and protection, installation, backup, system support	3 days

Figure 1-1. Ada Programming Support Environment Curriculum Modules

NO.	TITLE	DESCRIPTION	DURATION
L101	Ada Orientation for Managers	overview of development and features of Ada	1/2 day
L102	Ada Technical Overview	overview of language-introduction to language features in more depth than L101	1 day
L103	Introduction to High Order Languages	key HOL concepts for assembly language programmers	1 day
L104	Beginning Programming	introduction to computer programming in an Ada context	4 weeks
L201	Ada for Technical Managers	use of Ada for good systems design; packages, types, generics, portability features, etc.	3 days
L202	Basic Ada Programming	essentially the Pascal subset	1 week
L301	Using the Ada Language Reference Manual	how to use the ALRM effectively as a reference	2 days
L302	Use of Ada for Requirements	Ada as a requirements definition language	2 days
L303	Real Time Concepts	real time design concepts for technical managers	1 day
L304	Ada Reader's Course	reading an Ada design or program for its key points and overall structure	1 day
L305	Algorithms and Data Structures in Ada	packages, access types, private types, discriminated records, generics, basic tasking, basic algorithms	1 week
L401	Real Time Systems in Ada	everything about tasking, external interfaces, low-level features	1 week
L500	Specialty Courses	numerical analysis, hardware diagnostics, man/machine interface, database management, etc.	varying

Figure 1-2. Ada Language Curriculum Modules

NO.	TITLE	DESCRIPTION	DURATION
M101	Software Engineering for Managers	software life-cycle, top-down concepts, documentation, testing	1 day
M102	Introduction to Software Engineering	life-cycle, top-down concepts, overview of various methodologies	2 days
M201	Software Engineering Methodologies	thorough coverage of major methodologies	1 week
M202	Overview of a Specific Methodology	overview of an organization's selected life-cycle methodology	1/2 day
M301	Requirements Methodology	requirements definition techniques and methodology	1 week
M302	Design Methodology	how to do design with required methodology	4 days
M303	Coding Methodology	structured programming, coding standards, programming style, etc.	2 days
M304	Software Review Methodology	walkthroughs, code reading	1 day
M401	Introducing Ada to Your Organization	how to use the recommended curriculum to meet specific needs	1 day
M402	Psychological Aspects of Retraining	techniques for overcoming resistance to change	1 day

Figure 1-3. Software Engineering Curriculum Modules

This page intentionally left blank.

SofTech Model Ada Training Curriculum

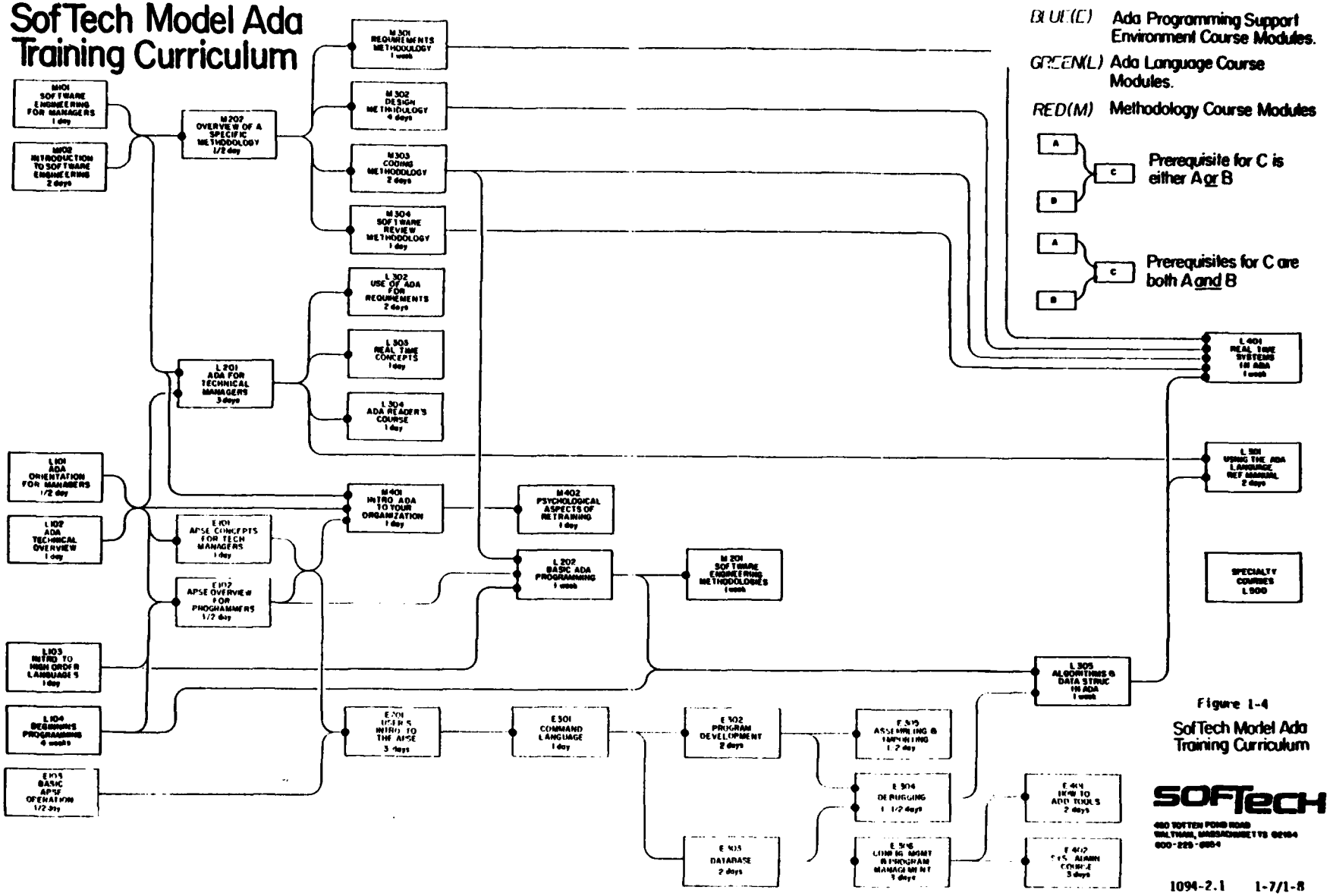


Figure 1-4

SofTech Model Ada Training Curriculum

SOFTECH
460 TOTTEN POND ROAD
WALTHAM, MASSACHUSETTS 02154
800-225-6864

SofTech Model Ada Training Curriculum

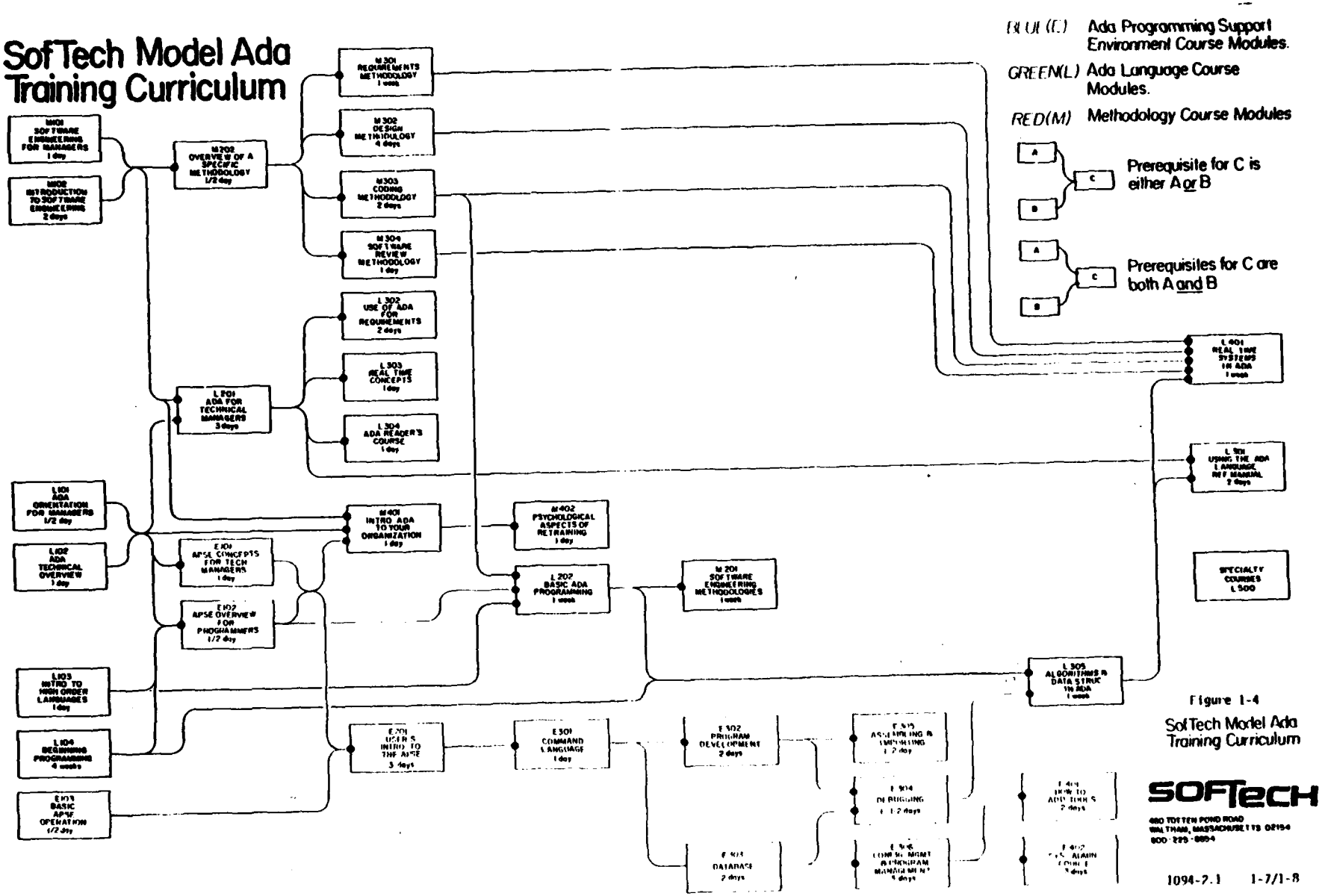


Figure 1-4

SofTech Model Ada Training Curriculum

SOFTECH
460 TOTTEN POND ROAD
WALTHAM, MASSACHUSETTS 02154
800-225-8604

Section 2

DESCRIPTION OF CONTRACT EFFORT

2.1 Introduction

The Ada Software Design Methods Formulation contract was performed by the Federal Systems Group of SofTech, Inc. under contract to the Department of the Army Communications-Electronics Command (CECOM). The contract was part of a CECOM effort intended to identify effective approaches to the use of Ada in designing and developing software for embedded computer systems. A previous solicitation awarded two contracts to redesign portions of existing embedded systems using Ada as a design language and to implement one selected module of each system in Ada. The Data Systems Division of the General Dynamics Corporation was charged with redesigning the AN/TYC-39 message switch; the Government Systems Division of the Control Data Corporation redesigned the Missile Minder AN/TSQ-73. The General Dynamics and Control Data contracts were performed simultaneously over one year. SofTech began work two months into the redesign efforts and continued for one year. SofTech's role was to provide Ada expertise to the designers, to interact with them in order to formulate case studies illustrating effective Ada design methods, and to recommend an overall approach to Ada training, to understand and assess difficulties which will be encountered in the use of Ada, and to understand how Ada and software design methods can be used together (see Figure 2-1).

2.2 Technical Approach

SofTech's technical approach for the Ada Software Design Methods Formulation contract was based upon close interaction with the government and the design contractors to ensure results consistent with government objectives. The contract comprised three principal tasks:

- Technical Interchange Forums with each design contractor and the extraction of case studies from the material presented at these meetings.

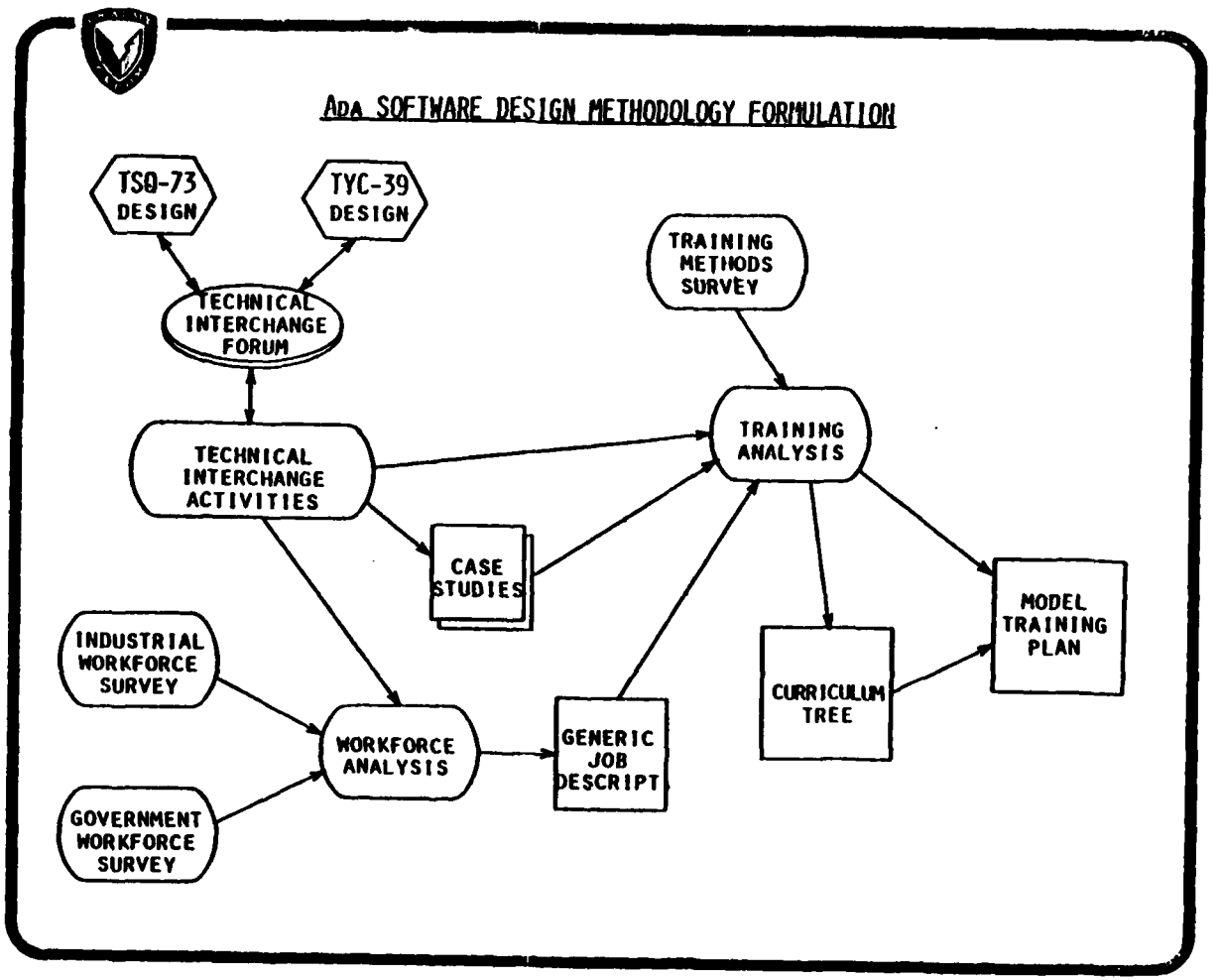


Figure 2-1. Project Overview

- Ada training requirements analysis.
- CECOM coordination.

The tasks are described in the following subsections.

2.2.1 Ada Program Design Technical Interchange

SofTech interacted with the design contractors in order to observe and document Ada usage issues as well as to formulate Ada design methods in response to these issues. Softech's technical interchange task was conducted by a team with extensive experience in Ada as well as an understanding of government embedded software applications. Dr. John Goodenough, SofTech's Director of Research and Development and an Ada Distinguished Reviewer and Mr. Nico Lomuto, Program Manager of the Ada Compiler Validation Capability effort served as SofTech's Ada experts during the technical interchange task; Dr. Sterling Eanes, Director of SofTech's Advanced Development Group was SofTech's authority on embedded software systems. Other members of the SofTech team were Christine Ausnit, Christine Braun, John Doyle, Karen Sather and Putnam Texel. This task involved two subtasks, the Technical Interchange Forum and the Case Studies, which are detailed below.

2.2.1.1 Technical Interchange Forum

Technical interchange meetings were held on a regular basis with both design contractors and provided the opportunity for a mutual exchange of information. The meetings, chaired by CECOM, occurred on the following dates:

General Dynamics	Control Data
December 1 and 2, 1981	December 16 and 17, 1981
February 18 and 19, 1982	January 20 and 21, 1982
April 13 and 14, 1982	March 11 and 12, 1982
May 25 and 26, 1982	April 15 and 16, 1982
July 14, 1982	June 2, 1982
September 1 and 2, 1982	July 13, 1982

The format of these meetings was typically informal, and consisted of a continuing review of the design contractors' approaches, and of their difficulties and successes in using both a particular design methodology and Ada as a design language. In addition to the group interchanges, time was often set aside for one-to-one, non-judgemental discussions between SofTech and the design teams.

SofTech's role in the meetings was to address any Ada-specific questions and to extract material for case studies of general interest to the embedded system community. From the outset, SofTech had identified the following objectives for these meetings:

- To understand each design contractor's use of Ada as a design language and any difficulties they had using Ada.
- To gain information about the relation between programmer skill levels and difficulties in using Ada effectively.
- To provide suggestions to the design contractors on the use of Ada for design.
- To understand and document which Ada concepts pose the most difficulty when using Ada for design.
- To understand the systems being redesigned and the design approaches of both contractors.
- To promote the discussion of issues which would prove useful in case studies of design issues.
- To elicit information helpful in recommending Ada training approaches and materials.

Given these objectives, attention at the Technical Interchange Forum was devoted to topics like the following:

- What is the appropriate structure for the system being designed? In particular, how can Ada be used to improve the system's maintainability without imposing an unacceptable reduction in run-time efficiency?
- How well does each design contractors' methodology fit with the use of Ada to document the evolving design?
- What Ada coding standards would help to make the design more understandable and maintainable?

- In what ways does Ada appear to be unsuitable for use as a design language, and how should these difficulties be resolved?

Under the terms of the contract, SofTech prepared and distributed minutes of all technical interchange meetings. (These minutes are not included as part of the present report.)

While SofTech's detailed observations on the design efforts can be found in the Case Studies Report, some general comments on our findings are perhaps appropriate here. First, it was apparent throughout the contract that Ada tasking was attracting the greatest attention. The effective use of a feature so new and so powerful is difficult in the absence of paradigms. Surprisingly, even the conceptual difference between tasks, procedures and packages was found difficult to grasp in certain cases. Several specific questions were identified, for which explicit guidelines would be desirable. For instance, how does a designer determine the correct tasking structure for a given system design and integrate this structure with a given methodology? And what are the performance implications of a given tasking structure? One might speculate that the actual problem is the absence of real-time design paradigms in general, and that Ada tasking finally offers a notation within which such paradigms might be presented.

Second, the role of the runtime system in the design needs to be clarified. Taking as typical examples the issues of distributed tasking and (system-dependent) fault detection and recovery, what does the language guarantee? What does it permit but not guarantee? Should one base the design on a particular implementation or should one procure a customized runtime system? In the case of unique hardware, at which phase in the design should the runtime system be specified, and by whom?

Third, Ada was found to be very effective in stating system requirements, particularly hardware interfaces and timing constraints. (It was interesting in this context, that through a rigorous specification of requirements in Ada, one contractor saved a significant amount of time in the detailed design and coding phases.)

Fourth, the use of packages requires guidelines, as does the use of exceptions and types. With exceptions for instance, should they be raised to cause a deviation in the flow of control or strictly for program errors? How should hardware exceptions and impossible states be handled? As for types, at what stage should they enter into a design, and with what abundance? How is the proliferation of types to be avoided?

Fifth, given the fact that reusable software modules are a key objective of the Ada program, how should these modules be specified and promoted?

Sixth, the design efforts brought out the need for a number of automated tools, such as System Design Language (SDL) and Program Design Language (PDL) support, data dictionaries, a cross-reference capability, editors, and tools for maintaining charts and diagrams and for performing program structure analysis.

Finally, it was obvious that the user community needs paradigms for solving common design problems with Ada and that Ada training must stress the practical use of Ada as well as language constructs. (The reader is referred to Appendix A of the Case Studies Report for a more in-depth discussion of areas for future research.)

2.2.1.2 Case Studies

As part of the Ada Program Design Technical Interchange task, SofTech developed case studies illustrating in detail the effective use of Ada to solve the kinds of design problems that arise in developing embedded software systems. The case studies are intended to suggest possible guidelines for the use of Ada as a design language by providing examples of the proper use of Ada in representative design problems. Further, the case studies provide concrete examples supporting SofTech's curriculum recommendations. (Figures 5-5, 5-6, and 5-7 cross-reference recommended course modules by case studies.)

Information for the case studies was obtained from the technical interchange meetings with each design contractor and from analysis of the design contractors' final reports. The case studies were selected to illustrate:

- the conceptual difficulties faced by novice Ada users (see discussion in Section 2.2.1.1)

- guidelines for the use of Ada discovered by the design contractors and SofTech, and concrete illustrations of how the guidelines are to be applied (See Case Studies Report, Sections 2.3, 2.4, 3.2, 3.4, 3.6, 3.7, 3.8, 3.9, 4.4, 5.2, and 6.1; see also the following case studies: Task Structure for a Target Tracking System, UART: Expressing Hardware Design in Ada, Tasks and Structure Charts, Stubbing and Readability, Succintness of Range Constraint, Memory-mapped I/O in Ada, and Eliminating GOTOs.)

- the process by which a good design evolves and how Ada did (or can) help to develop a good design (See Case Studies Report, Sections 2.3, 2.4, 3.2, 3.3, 3.4, 3.6, 3.8, 3.9, 4.3, 4.4, 6.3; see also the following case studies: Power Failure Requirements, Functional Description of an Air Defense System, and Task Structure for a Target Tracking System.)

- the ease or difficulty of using Ada as a design language in the context of the different design methodologies chosen by the design contractors (See Case Studies Report, Sections 2.2, 2.4, 2.5, 3.2, 3.4, 3.9, 4.3, 4.4, 6.1, 6.3, 6.4, 6.5, 6.6; see also the following case studies: Power Failure Requirements and Use of Types to Describe Hardware Interface Requirements.)

- the interaction between the use of Ada as a design language and the particular design methodology chosen by the design contractors (See Case Studies Report, Sections 2.2, 2.4, 3.4, 3.9, and 4.3.).

The case studies are presented in a uniform, self-contained format. Each begins with a background section which states the case study objective, describes the designer's problem, and gives a brief discussion of the problem. The detailed example then follows. It includes a problem statement, a solution outline, and a detailed solution. Each case study concludes with an epilogue that discusses what was learned and how it might be applied.

The Case Studies Report is submitted as a separate deliverable item under the present contract.

2.2.2 Training Requirements Analysis

The purpose of the Training Requirements Analysis task was threefold:

- To propose a general job classification schema for personnel working on large-scale embedded systems.
- To identify the Ada skills required by each job category.
- To recommend a model training curriculum to meet the needs of the embedded systems work force.

The following subparagraphs describe SofTech's approach to the Training Requirements Analysis task.

2.2.2.1 Industry/Government Work Force Survey

In order to formulate a generic hierarchy of job categories for the embedded systems work force, SofTech administered the Industry/Government Work Force Survey to ten companies and government facilities selected by CECOM. (The names of survey participatns are given in Appendix A; the Industry/Government Work Force Survey appears as Appendix B.)

The survey was designed by SofTech with the assistance of Cece Menkin, an independent survey consultant. The questions were structured so as to elicit forced responses in the following areas:

1. Technical Background
2. Functional Job Description
   - outputs
   - principal duties
   - general activities
3. Familiarity with
   - programming languages
   - methodologies/system design concepts
   - programming concepts and constructs
   - Ada concepts

(Question A-9, regarding the fictitious "Beamson Tables," was added as a control.) It was thought that these responses would provide the raw data for an analysis of the work force, and indeed, this proved to be the case.

The general job classification (see Section 3) was arrived at by statistically clustering similar responses to survey questions, each significant cluster constituting a node in the classification schema. Mr. Robert Muller of MIT performed the statistical analysis for SofTech using the Consistent System under MULTICS. A complete description of the clustering technique used by Mr. Muller can be found in Appendix D.

It must be emphasized that the purpose of the work force survey and the resulting general job classification was to understand what functions are performed by personnel working on the development of large-scale, embedded systems; each job category was characterized by typical capabilities, duties, and technical and academic backgrounds for an employee in that category. The name of each of the general job categories should be considered simply an abbreviation or name tag for the functions which would be performed by an employee holding that job.

2.2.2.2 Curriculum Tree

Once the clustering of survey responses had established generic job classification, SofTech prepared a curriculum tree. The initial idea for the curriculum tree was suggested by the Statement of Work for the contract (see Figure 3-3). Essentially, the tree is a mechanism for representing graphically the background and "Ada viewpoint" of each job category as well as the functional organizational dependencies interrelationships between categories. The background for each category was determined by analyzing data from the Industry/Government Work Force Survey according to the following criteria:

1. Years of Software Development of Support
2. Principal Output and Duties
3. Programming Languages (Studied or Used)
4. Knowledge of Methodologies

5. Knowledge of Programming Constructs

6. Knowledge of Programming Concepts

7. Knowledge of Ada Programming Concepts

8. Self Described Titles

The background was then analyzed in conjunction with the job description for each functional category, whereupon SofTech made an assessment of the "Ada Viewpoint" (i.e., knowledge requirements) necessary for the job. The Ada Viewpoint was seen by SofTech as encompassing the Ada language, the Ada Programming Support Environment, and software engineeering methodologies. The curriculum tree is presented in Section 4.

2.2.2.3 Model Ada Training Program

Based on the Ada knowledge requirements identified in the Curriculum Tree, SofTech has recommended a comprehensive Model Ada Training Program. This curriculum is divided into three principal areas -- the Ada Programming Support Environment, the Ada language, and software engineering methodologies (see Figures 1-1, 1-2, and 1-3). Detailed Module Descriptions for the Model Ada Training Program are presented in Section 5.2. Course modules from each area are recommended in varying combinations for each job category, depending upon Ada knowledge requirements (see Section 5.3).

2.2.2.4 Industrial Training Survey

In order for SofTech's recommended Model Ada Training program to achieve the goals of the Army's Ada Program, its implementation must address two questions. First, what is the general feeling in industry regarding the effectiveness of various training approaches? And second, what resources are presently in place to support Ada training? As part of the Ada Software Design Methods Formulation effort, SofTech designed and administered the Industrial Training Survey to query industry on these as well as other related issues. The survey was given to six corporations, each with extensive involvement in large-scale embedded software projects. The results of the survey are presented in Section 6.

2.2.3 CECOM Liaison

The purpose of this task was to ensure that the government's concerns and priorities were represented in the development of the Ada case studies and model training program, and to assist in the Industry/ Government Work Force Survey. To perform this task, SofTech established a Ft. Monmouth office staffed with two full time individuals. They provided day-to-day coordination with the government to ensure that the program best served the government's needs. Tasks included:

1. Participating in and reviewing the findings of SofTech's technical interchange activities with the government.

2. Discussing evolving observations on effective Ada design methods with government personnel involved in Ada developments.

3. Contributing to the development of the survey questionnaires and ensuring that survey instruments were consistent with government objectives.

4. Working with the government to ensure that the Curriculum Tree reflects government objectives.

5. Interacting with the government to determine training concerns and priorities.

Section 3

INDUSTRY/GOVERNMENT WORK FORCE SURVEY

3.1 Survey Development and Distribution

The Industry/Government Work Force Survey was administered to ten companies and government agencies selected by CECOM. (See Appendix A for a list of participating companies and government agencies.) The following sections describe the purpose of the Industry/ Government Work Force Survey, development of survey questions, and the survey distribution and administration.

3.1.1 Purpose

The purpose of the Industry/Government Work Force Survey was to provide a functional decomposition of the industrial and government embedded computer systems work force.

3.1.2 Development of Survey Questions

Design of the survey instrument addressed the need to solicit the following information from each respondent:

- Technical background (questions 1, 2, 5, and 8)
- Principal outputs and duties (questions 6, 7, 9, 10, and 11)
- Knowledge of programming languages, software methodologies, programming concepts and constructs and Ada concepts (questions 16, 17, and 18)
- Miscellaneous information (questions 2, 3, 4, 12, 13, 14, 15, 19, and 20).

Ms. Cece Menkin served as a consultant during the formulation of the survey instrument. Questions were constructed with a "forced response" format to facilitate completion of the survey by the respondent and to assist in tabulation of the responses. Open ended questions were deliberately not used. A control feature incorporated in the survey was the listing of non-existent "Beamson tables" in a section which called for a response indicating the respondent's level of knowledge (question 18A - 9).

3.1.3 Survey Distribution and Administration

A meeting was held on February 23, 1982 at CECOM headquarters Ft. Monmouth, New Jersey for all survey participants (see Appendix A for list). At this meeting the general purpose of the survey was discussed and participants had the opportunity to respond to each item of the survey. Changes in the survey which resulted from this meeting were incorporated and the surveys were distributed on March 1, 1982. The final version of the survey is found in Appendix B.

A key element in the survey administration process was a primary contact at each company. This person was charged with distributing the surveys, following up on tardy respondents, and ensuring the return of the set of completed questionnaires to SofTech. The selection of the primary contact for each company was made at the February 23rd meeting.

3.2 Survey Findings

A total of 428 Industry/Government Work Force Surveys were returned out of a distribution of 720 (59% returned). Figure 3-1 shows summarized source data indicating the percentage of responses for each survey question. These percentages are calculated on the number of surveys returned.

Figure 3-2a ranks response percentages from question 17, in which respondents listed the two programming languages with which they are most proficient. Figure 3-2b ranks the summed percentages of respondents indicating either moderate or frequent usage of methodologies listed in question 18A. Figure 3-2c ranks the summed percentages of respondents indicating either moderate or frequent usage of programming constructs listed in question 18B. Figure 3-2d ranks the summed percentages of respondents indicating either moderate or frequent usage of programming concepts listed in question 18C. Figure 3-2e ranks the summed percentages of respondents indicating either moderate or frequent usage of Ada programming concepts listed in question 18D.

ADA SOFTWARE DESIGN METHOD FORMULATION
INDUSTRY/GOVERNMENT WORK FORCE SURVEY

1. How many years have you been involved with software development and/or support?
   a. 0-2 years 9.7% b. 2-5 years 23.9% c. 5-10 years 20.1% d. Over 10 years 46.3%

2. Which of the following areas describe your experience with software development and/or support?
   a. Commercial 34.6%
   b. Military 80.1%
   c. Embedded Computer Systems 40.4%
   d. Educational 11.9%
   e. Statistical 5.8%
   f. Other 7.5%

3. To date, what has been your level of involvement with Ada?
   a. Do not know what Ada is. 3.8%
   b. Have heard of Ada but am not familiar with Ada concepts. 52.9%
   c. Participated in orientation sessions about Ada. 23.8%
   d. Have had Ada training. 8.2%

4. If you have had Ada training, what type of training was it?
   a. Video tape 8.5%
   b. College course 3.7%
   c. In-house course 24.3%
   d. Programmed learning 2.6%
   e. Seminar 22.8%
   f. Informal, on the job training 27.0%
   g. Other (Explain) 11.1%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 1 of 12)

DEVELOPMENT

5. How many years have you worked on the development of large-scale, embedded computer systems?

a. 0-1 year 20.0% b. 1-3 years 20.0% c. 3-5 years 16.7% d. Over 5 years 43.3%

IF YOUR DUTIES ARE PRINCIPALLY IN THE DEVELOPMENT AREA, PLEASE ANSWER THE FOLLOWING QUESTIONS:

6. What outputs do you produce? (Check as many as appropriate.)

29.2% a. Hardware/software tradeoff evaluation
46.0% b. Data flow diagrams
35.0% c. Test drivers
67.1% d. Code
61.2% e. Program design language or flow charts
52.1% f. Requirements specifications
59.6% g. Design specifications
55.1% h. Test plans
38.6% i. Integration plans
23.1% j. Management plans
24.3% k. Cost data
25.9% l. Analysis reports/summaries
40.4% m. Milestone charts/schedules
52.1% n. Status reports
27.1% o. Interview sheets/Hiring recommendations
21.5% p. Correspondence
3.3% q. Other - (Explain)

7. Which of the following describe your principal duties?

a. Requirements/Analysis Review - (22.0% Conduct, 34.3% Attend)
45.1% b. System Analysis
63.3% c. Design
d. Design Review (35.3% Attend, 29.9% Attend)
54.9% e. Code
f. Structured Walkthroughs (25.2% Conduct, 29.9% Attend)
10.7% g. Formulation of Policy
14.0% h. Formulation of Strategy
27.6% i. Technical Management
7.2% j. Program Management
7.2% k. Configuration Management
7.2% l. Quality Assurance
5.3% m. Monitoring contracts
2.3% n. Other (explain)

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 2 of 12)

8. How many years have you worked on the support of large-scale, embedded computer systems?

   a. 0-1 year 33.1% b. 1-3 years 25.4% c. 3.5 years 14.6% d. Over 5 years 26.8%

IF YOUR DUTIES ARE PRINCIPALLY IN THE SUPPORT AREA, PLEASE ANSWER THE FOLLOWING QUESTIONS:

9. What outputs do you produce? (Check as many as appropriate.)

22.7%	a.	Software trouble report analyses	5.6%	j.	Updated training manuals
10.7%	b.	Temporary (proposed) Engineering Change Proposals	16.6%	k.	Updated user manuals
12.2%	c.	Red lined documentation	19.9%	l.	Software Trouble Reports (STRs)
16.4%	d.	Test plans	4.7%	m.	Automated build systems
9.3%	e.	Test drivers	7.5%	n.	Management information reports
9.3%	f.	Technical advice to Configuration Control Board	4.7%	o.	Version description documents
4.7%	g.	Updated MIL-STD specification	2.6%	p.	Version audits
5.8%	h.	Library Control	1.9%	q.	Field engineering reports
4.5%	i.	Maintain configuration procedures	3.7%	r.	Other (explain)

10. Which of the following describe your principal duties? (Check as many as appropriate.)

23.6%	a.	Analysis	3.0%	h.	Program Management
20.1%	b.	Design	8.4%	i.	Software Configuration Control Board participation
	c.	Design Review (10.7% Conduct, 17.3% Attend)			
20.6%	d.	Code/Patch	4.7%	j.	Configuration management
	e.	Structured Walkthroughs (8.2% Conduct, 12.2% Attend)	5.8%	k.	Quality Assurance
10.3%	f.	Technical Management	4.4%	l.	Monitoring contracts
4.0%	g.	Formulation of policy	3.0%	m.	Other (explain)

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 3 of 12)

11. RATE THE IMPORTANCE OF THE FOLLOWING ACTIVITIES AS THEY APPLY TO YOUR PRESENT JOB. REVIEW ALL CATEGORIES BUT CONSIDER ONLY THOSE ACTIVITIES WHICH YOU PERFORM AS PART OF YOUR RESPONSIBILITIES.

	PRIMARY	SECONDARY	MARGINAL	N/A (or no response)
A. MANAGEMENT/ADMINISTRATIVE				
1. Program management	7.0%	8.9%	9.6%	74.5%
2. Sales/marketing	1.9%	5.6%	9.3%	83.2%
3. Contract negotiation	1.2%	4.4%	10.7%	83.7%
4. Formulating policy	6.3%	7.5%	11.2%	75.0%
5. Formulating strategy	10.0%	10.5%	13.1%	66.4%
6. Preparing budgets/cost estimates	13.8%	17.3%	14.3%	54.6%
7. Technical management	28.3%	11.0%	8.6%	52.1%
8. Interviewing personnel	7.9%	11.4%	21.0%	59.7%
9. Preparing and revising schedules	15.9%	22.9%	17.5%	43.7%
10. Preparing management information reports	11.0%	18.2%	12.9%	57.9%
11. Preparing field engineering reports	0.2%	3.5%	6.1%	90.2%
12. Other administrative tasks	4.6%	12.4%	15.7%	67.5%
B. CONFIGURATION/QUALITY CONTROL				
1. Giving technical advice to configuration control board	7.7%	11.7%	13.6%	67.0%
2. Maintaining configuration procedures	4.7%	10.3%	15.7%	69.3%
3. Library control	2.8%	7.5%	13.3%	76.4%
4. Preparing version audits	2.8%	3.5%	7.7%	86.0%
5. Quality Assurance	5.8%	8.4%	14.5%	71.3%
6. Preparing temporary (proposed) engineering change reports/requests	4.7%	12.6%	13.1%	69.6%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 4 of 12)

	PRIMARY	SECONDARY	MARGINAL	N/A (or no response)
C. *EDUCATION/SELF-DEVELOPMENT*				
1. Preparing technical reports or papers	10.5%	21.7%	22.4%	45.4%
2. Reading technical magazines, papers, etc.	12.9%	34.1%	29.0%	24.0%
3. Reviewing technical work of others	23.8%	33.4%	15.4%	27.4%
4. Teaching others (including preparation)	10.3%	24.5%	29.0%	36.2%
5. Updating training manuals	2.6%	7.9%	18.7%	70.8%
6. Being trained (including preparation)	9.8%	19.4%	31.8%	39.0%
D. PROGRAM DESIGN/IMPLEMENTATION				
1. Functional system design (or architecture)	45.1%	15.7%	14.5%	24.7%
2. Functional module or subsystem design	50.2%	16.8%	10.3%	22.7%
3. Defining global data structures	32.0%	23.4%	13.8%	30.8%
4. Defining subsystem (module) interfaces	36.7%	24.1%	13.6%	25.6%
5. Defining data structures and algorithms for your own use	40.7%	18.2%	12.6%	28.5%
6. Coding	45.6%	15.2%	13.6%	25.6%
7. Debugging or modifying code	47.7%	14.3%	13.1%	24.9%
E. DOCUMENTATION				
1. Originating system requirements documents	22.4%	20.1%	16.1%	41.4%
2. Updating MIL-STD specifications	6.1%	11.0%	11.7%	71.2%
3. Preparing red lined documentation	9.6%	13.1%	14.7%	62.6%
4. Preparing version description documents	6.8%	11.7%	19.9%	61.6%
5. Originating/updating user manuals	10.3%	25.2%	21.3%	43.2%
6. Documenting code	32.9%	25.2%	9.8%	32.1%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 5 of 12)

	PRIMARY	SECONDARY	MARGINAL	N/A (or no response)
F. TESTING				
1. Defining system test cases	20.3%	23.6%	15.2%	40.9%
2. Preparing test drivers	13.6%	22.0%	17.1%	47.3%
3. Preparing test plans	23.1%	23.1%	16.8%	37.0%
4. Hardware testing	5.4%	9.6%	18.5%	66.5%
5. System software testing	37.9%	22.9%	12.4%	26.8%
6. Defining module test cases	26.9%	19.9%	17.3%	35.9%
7. Software module testing	35.5%	18.9%	12.9%	32.7%
8. Documenting test results	17.5%	26.2%	18.2%	38.1%
9. Preparing software trouble/discrepancy reports	15.4%	19.9%	20.1%	44.6%
10. Analyzing software trouble reports	19.9%	20.8%	15.7%	43.6%

12. Do you belong to any technical societies or working groups outside of your company? Yes 34% No 66%

13. During the past year, have you attended any job related conferences? Yes 41% No 59%

14. Have you ever published or presented a paper? Yes 23% No 78%

15. Do you read job-related magazines or newsletters?

a. Regularly 44.0%
b. Occassionally 44.5%
c. Only as my job demands 7.6%
d. Never 3.3%
e. Other (explain) 0.5%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 6 of 12)

KNOWLEDGE OF ALL LANGUAGES

16. Which of the following programming languages have you studied or used? Please check all that apply.

20.8%	a. JOVIAL	21.5%	j. ALGOL	17.5%	s. SNOBOL
31.5%	b. CMS-2	22.2%	k. RATFOR, WATFOR, WATFIV	0.7%	t. ECL
8.4%	c. C	1.9%	l. MODULA	14.5%	u. GPSS
93.2%	d. FORTRAN	2.8%	m. SIMULA	1.6%	v. SAS
48.1%	e. COBOL	2.6%	n. XPL	0.2%	w. PROTEGE
88.3%	f. ASSEMBLER	0.0%	o. MMP	0.0%	x. PPL
46.5%	g. PL-1	7.2%	p. FORTH	26.2%	y. APL
48.8%	h. PASCAL	25.0%	q. Ada	22.2%	z. Other (specify)
64.7%	i. BASIC	15.4%	r. LISP		

17. Two most proficient programming languages:

3.5%	a. JOVIAL	1.2%	j. ALGOL	0.0%	s. SNOBOL
10.3%	b. CMS-2	0.0%	k. RATFOR, WATFOR, WATFIV	0.0%	t. ECL
0.2%	c. C	0.0%	l. MODULA	0.7%	u. GPSS
49.1%	d. FORTRAN	0.0%	m. SIMULA	0.0%	v. SAS
8.4%	e. COBOL	0.0%	n. XPL	0.2%	w. PROTEGE
50.0%	f. ASSEMBLER	0.0%	o. MMP	0.0%	x. PPL
9.8%	g. PL-1	0.9%	p. FORTH	3.0%	y. APL
18.0%	h. PASCAL	0.9%	q. Ada	7.2%	z. Other (specify)
9.8%	i. BASIC	0.5%	r. LISP		

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 7 of 12)

18. INDICATE THE LEVEL OF KNOWLEDGE YOU HAVE WITH THE FOLLOWING METHODOLOGIES.

A. Methodology	(No response) or Unfamiliar	Have Heard of But Do Not Understand	Know What Concept Is	Have Used to A Moderate Extent	Have Used Frequently
1. PSL/PLA	64.7%	11.2%	20.6%	3.5%	0.0%
2. SADT	81.5%	5.4%	8.4%	4.2%	0.5%
3. SREM	85.9%	5.1%	7.9%	0.9%	0.2%
4. HIPO	30.4%	11.4%	29.9%	21.3%	7.0%
5. Jackson Design	78.4%	7.9%	8.4%	3.0%	2.3%
6. Structured Design	8.4%	3.3%	17.5%	27.6%	43.2%
7. Warnier/Orr Design	73.6%	9.1%	11.9%	4.2%	1.2%
8. N-S/Chapin Chart	76.4%	8.9%	9.3%	4.0%	1.4%
9. Beamson Tables	96.3%	2.1%	1.4%	0.2%	0.0%
10. Program Design Language	15.6%	8.2%	24.3%	22.9%	29.0%
11. Structured Programming	0.9%	1.6%	12.4%	24.1%	61.0%
12. Structured Walkthroughs	11.1%	4.7%	28.3%	29.0%	26.9%
13. Top-Down Design	2.1%	1.4%	12.6%	28.3%	55.6%
14. Top-Down Testing	7.0%	3.5%	26.2%	27.6%	35.7%
15. Bottom-up Design	14.5%	4.2%	39.7%	25.5%	16.1%
16. Bachman Diagramming	89.9%	3.5%	5.4%	0.5%	0.7%
17. Entity Diagrams	89.8%	2.8%	5.1%	1.6%	0.7%
18. Data Abstraction	55.9%	9.3%	18.0%	10.5%	6.3%
19. Other (specify)	--	0.0%	0.5%	1.2%	1.9%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 8 of 12)

B. Programming Constructs	(No response) or Unfamiliar	Have Heard of But Do Not Understand	Know What Concept Is	Have Used to A Moderate Extent	Have Used Frequently
1. Enumeration types	47.0%	6.5%	16.8%	15.9%	13.8%
2. Floating point types	3.5%	1.6%	17.1%	29.9%	47.9%
3. Fixed point types	8.8%	1.2%	10.7%	24.8%	54.5%
4. User defined types	16.2%	4.7%	20.6%	26.0%	32.5%
5. Pointers	7.4%	1.4%	15.0%	24.8%	51.4%
6. Typed pointers	29.2%	8.9%	22.4%	15.2%	24.5%
7. Ranges	16.1%	3.5%	22.7%	24.3%	33.4%
8. Records	7.6%	2.3%	16.8%	25.2%	48.1%
9. Variant records	33.4%	8.2%	20.8%	15.9%	21.7%
10. Object/type declarations	19.2%	4.2%	17.5%	24.5%	34.6%
11. Global variables	2.9%	1.4%	7.9%	21.9%	65.9%
12. Local variables	3.5%	1.2%	7.2%	20.6%	67.5%
13. Formal and actual parameters	16.6%	4.4%	9.8%	18.5%	50.7%
14. Reserved words	7.6%	2.1%	13.6%	19.9%	56.8%
15. Blocks	11.5%	1.6%	15.2%	22.9%	48.8%
16. Case statements	8.6%	2.6%	12.2%	23.6%	53.0%
17. If/then/else statements	2.2%	0.5%	17.7%	21.8%	67.8%
18. Loop/for/while/until statements	2.8%	0.5%	9.8%	22.9%	64.0%
19. Exit Statements (for loops)	4.7%	0.7%	17.8%	28.7%	48.1%
20. Procedures	3.7%	2.1%	8.2%	22.2%	63.8%
21. Functions	3.0%	1.2%	7.5%	25.2%	63.1%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 9 of 12)

B. Programming Constructs (Continued)	(No response) or Unfamiliar	Have Heard of But Do Not Understand	Know What Concept Is	Have Used to A Moderate Extent	Have Used Frequently
22. Return statements	2.3%	0.7%	7.5%	18.7%	70.8%
23. Clusters/modules/ packages	15.3%	6.5%	17.5%	24.5%	36.2%
24. Stubs	19.1%	4.0%	18.9%	24.1%	33.9%
25. Goto statements	3.4%	0.2%	15.6%	31.5%	49.3%
26. Comments	3.1%	0.2%	3.0%	18.2%	75.5%
27. Exception handlers	18.2%	5.6%	22.9%	22.2%	31.1%
28. Task/coroutines	22.4%	6.1%	26.4%	18.2%	26.9%
29. Other (specify)	--	0.0%	0.5%	0.5%	0.7%
C. Programming Concepts					
1. Importing/exporting names	67.8%	9.1%	10.5%	4.9%	7.7%
2. Data encapsulation (compools)	38.8%	11.2%	21.7%	14.3%	14.0%
3. Name scoping	55.2%	7.7%	15.2%	7.9%	14.0%
4. Name visibility	57.7%	9.3%	14.5%	7.5%	11.0%
5. Static & dynamic nesting	35.1%	13.3%	20.1%	14.0%	17.5%
6. Iteration	8.4%	3.0%	11.2%	25.5%	51.9%
7. Conditional statements	5.9%	1.6%	5.1%	20.6%	66.8%
8. Recursion	12.2%	3.3%	23.8%	32.2%	28.5%
9. Concurrency	27.3%	6.8%	28.0%	18.7%	19.2%
10. Strong typing	43.3%	7.2%	21.0%	13.3%	15.2%
11. Type conversion	26.2%	6.8%	18.9%	22.4%	25.7%
12. Data abstraction	36.9%	13.1%	25.7%	10.5%	13.8%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 10 of 12)

C. Programming Concepts (Continued)	(No response) or Unfamiliar	Have Heard of but Do Not Understand	Know What Concept Is	Have Used to A Moderate Extent	Have Used Frequently
13. Generics	50.0%	10.5%	22.7%	9.3%	7.5%
14. Loop invariants	47.4%	11.9%	17.8%	11.2%	11.7%
15. Parameter binding	48.6%	12.4%	19.6%	8.9%	10.5%
16. Version number	28.4%	4.7%	17.1%	18.0%	31.8%
17. Other (specify)	--	0.2%	0.0%	0.2%	1.2%
D. Ada Programming Concepts					
1. Enumeration types	55.4%	10.3%	24.1%	6.5%	3.7%
2. User-defined types	41.9%	9.3%	34.8%	7.0%	7.0%
3. Subtypes	53.8%	13.1%	24.5%	4.2%	4.4%
4. Derived types	59.8%	14.7%	19.4%	4.0%	2.1%
5. Real types	38.6%	7.2%	34.8%	7.5%	11.9%
6. Floating point types	33.8%	6.1%	37.9%	8.4%	13.8%
7. Fixed point types	35.8%	5.6%	36.4%	7.9%	14.3%
8. Record types	41.9%	8.2%	32.2%	7.2%	10.5%
9. Record types with discriminants	63.8%	13.3%	17.5%	3.3%	2.1%
10. Slices	74.1%	9.8%	11.9%	2.3%	1.9%
11. Aggregates	71.7%	10.5%	13.8%	2.1%	1.9%
12. Allocators	72.1%	8.9%	15.0%	2.6%	1.4%
13. Access types	64.5%	9.1%	19.8%	3.3%	3.5%
14. Overloading	67.3%	9.3%	19.4%	2.1%	1.9%
15. Packages	57.5%	10.0%	25.5%	4.2%	2.8%
16. Private types	62.9%	8.2%	24.5%	3.0%	1.4%
17. Scope	58.2%	7.5%	26.4%	2.8%	5.1%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 11 of 12)

D. Ada Programming Concepts (Continued)	(No response) or Unfamiliar	Have Heard of But Do Not Understand	Know What Concept Is	Have Used to A Moderate Extent	Have Used Frequently
18. Short circuiting	77.5%	8.2%	11.5%	1.6%	1.2%
19. Visibility	63.0%	9.3%	21.7%	3.0%	3.0%
20. Tasking	46.4%	12.9%	31.1%	3.3%	6.5%
21. Task types	58.2%	13.6%	23.6%	2.3%	2.5%
22. Rendezvous	67.6%	7.0%	21.5%	2.3%	1.6%
23. Entries	59.9%	7.9%	24.8%	3.0%	4.4%
24. Entry Families	75.0%	11.0%	11.0%	1.6%	1.4%
25. Separate compilation	44.4%	6.1%	34.3%	5.6%	9.6%
26. Exceptions	55.2%	7.7%	27.1%	5.1%	4.9%
27. Generic program units	66.4%	10.7%	18.9%	2.8%	1.2%
28. Instantiation	72.5%	9.1%	15.4%	1.6%	1.4%
29. Elaboration	77.7%	8.6%	10.9%	1.4%	1.4%
30. Context specification	72.9%	11.0%	13.8%	1.4%	0.9%
31. Information hiding	60.5%	8.9%	25.2%	2.1%	3.5%
32. Mutual recursion	71.7%	11.7%	13.8%	1.6%	1.2%
33. Other (specify)	--	0.2%	0.0%	0.0%	0.0%

Figure 3-1. Summarized Source Data Industry/Government Work Force Survey (Page 12 of 12)

PROGRAMMING LANGUAGES
RANKED LISTING OF RESPONDENT PROFICIENCY

LANGUAGE	PERCENT OF RESPONDENTS
ASSEMBLER	50.0
FORTRAN	49.1
PASCAL	18.0
CMS-2	10.3
BASIC	9.8
PLI	9.8
COBOL	8.4
Other	7.7
JOVIAL	3.5
APL	3.0
ALGOL	1.2
Ada	0.9
GPSS	0.7
LISP	0.5
PROTEGE	0.2
C	0.2
FORTH	0.2
XPL	0.0
SNOBOL	0.0
ECL	0.0
SIMULA	0.0
SAS	0.0
MODULA	0.0
RATFOR WATFOR WATFIV	0.0
MMP	0.0
PPL	0.0

Figure 3-2a. Programming Languages

METHODOLOGIES
RANKED BY SUMMED PERCENTAGES OF THOSE USED
FREQUENTLY OR MODERATELY

METHODOLOGY	PERCENT
Structured Programming	85.0
Top Down Design	83.9
Structured Design	70.8
Top Down Testing	63.3
Structured Walkthroughs	55.8
Program Design Language	51.9
Bottom Up Design	41.6
HIPO	28.3
Data Abstraction	16.8
Jackson Design	5.4
Warnier Orr Design	5.4
N S Chapin Chart	5.4
SADT	4.7
PSL PLA	3.5
other methodology	3.0
Entity Diagrams	2.3
Bachman Diagramming	1.2
SREM	1.2
Beamson Tables	0.2

Figure 3-2b. Methodologies

PROGRAMMING CONSTRUCTS
RANKED BY SUMMED PERCENTAGES OF THOSE USED
FREQUENTLY OR MODERATELY

CONSTRUCT	PERCENT
comments	93.7
return statements	89.5
if then else statements	89.5
functions	88.3
local variables	88.1
global variables	87.9
loop for while until	86.9
procedures	86.0
fixed point types	82.2
goto statements	80.8
floating point types	77.8
exist statements	76.9
case statements	76.6
reserved words	76.6
pointers	76.2
records	73.4
blocks	71.7
format actual params	69.2
clusters modules package	60.7
object type dcls	59.1
user defined types	58.4
stubs	58.0
ranges	57.7
exceptions handlers	53.3
task coroutines	45.1
typed pointers	39.5
variant records	37.6
enumeration types	29.7

Figure 3-2c. Programming Constructs

PROGRAMMING CONCEPTS
RANKED BY SUMMED PERCENTAGE OF THOSE USED
FREQUENTLY OR MODERATELY

CONCEPT	PERCENT
conditional statements	87.4
iteration	77.3
recursion	60.7
version number	49.8
type conversion	48.1
concurrency	37.9
static dynamic nesting	31.5
strong typing	28.5
data encapsulation	28.3
data abstraction	24.3
loop invariants	22.9
name scoping	22.0
parameter binding	19.4
name visibility	18.5
generics	16.8
importing exporting name	12.6
other prog concepts	1.4
other prog constructs	1.2

Figure 3-2d. Programming Concepts

ADA CONCEPT
RANKED BY SUMMED PERCENTAGE OF THOSE USED
FREQUENTLY OR MODERATELY

ADA CONCEPT	PERCENT
Ada float point types	22.2
Ada fixed pt types	22.2
Ada real types	19.4
Ada record types	17.8
Ada separate compilation	15.2
Ada user defined types	14.0
Ada enumeration types	10.3
Ada tasking	9.6
Ada subtypes	8.6
Ada scope	7.9
Ada entries	7.5
Ada packages	7.0
Ada access types	6.5
Ada visibility	6.1
Ada derived types	6.1
Ada information hiding	5.9
Ada rec types discrim	5.9
Ada task types	4.7
Ada private types	4.4
Ada slices	4.2
Ada overloading	4.0
Ada allocators	4.0
Ada rendezvous	4.0
Ada aggregates	4.0
Ada generic prog units	4.0
Ada entry families	3.0
Ada instantiation	2.8
Ada short circuiting	2.8
Ada elaboration	2.8
Ada mutual recursion	2.8
Ada context spec	2.3
other Ada concepts	0.0

Figure 3-2e. Ada Concepts

3.3 General Job Category Classification

The Industry/Government Work Force Survey was used as the primary vehicle for establishing general job classifications for persons working on large-scale embedded computer systems, along with the duties and representative background of each classification. (The classifications are detailed in Section 3.3.3.)

3.3.1 Job Classification Development Procedure

SofTech's analysis of the Industry/Government Work Force Survey used the process of interpreted nominal clustering, where respondents were grouped according to the similarity of their responses. (Appendix D describes the steps involved in interpreted nominal clustering.) As anticipated, the generic job categories identified by the clustering analysis were not totally consistent with those which appear in the Statement of Work under the discussion of The Curriculum Tree. (The categories originally identified in the Statement of Work are detailed in Figure 3-3.) The survey data initially revealed five clusters which SofTech designated as Senior Engineering Manager, Project Administrative Manager, Support Manager, Development Engineering, and System Integration Engineering (see Figure 3-4). Upon analyzing the principal duties and outputs of each of these clusters it became apparent that the Development Engineering and the System Integration Engineering clusters should be further tested for sub-clusters. As Figure 3-5 shows, Development Engineering was then divided into five subclusters (Project/Task Leader, Configuration Management/Quality Assurance Engineer, Design Consultant, Software Developer, and Junior Staff Member/Technical Aide) and System Integration Engineering was divided into three subclusters (System Integration Manager Research Staff, System Integration Senior Technical Staff, and System Integration Engineer).

Figure 3-3. Curriculum Tree from Government Statement of Work

Figure 3-4. Initial Job Classification Clusters

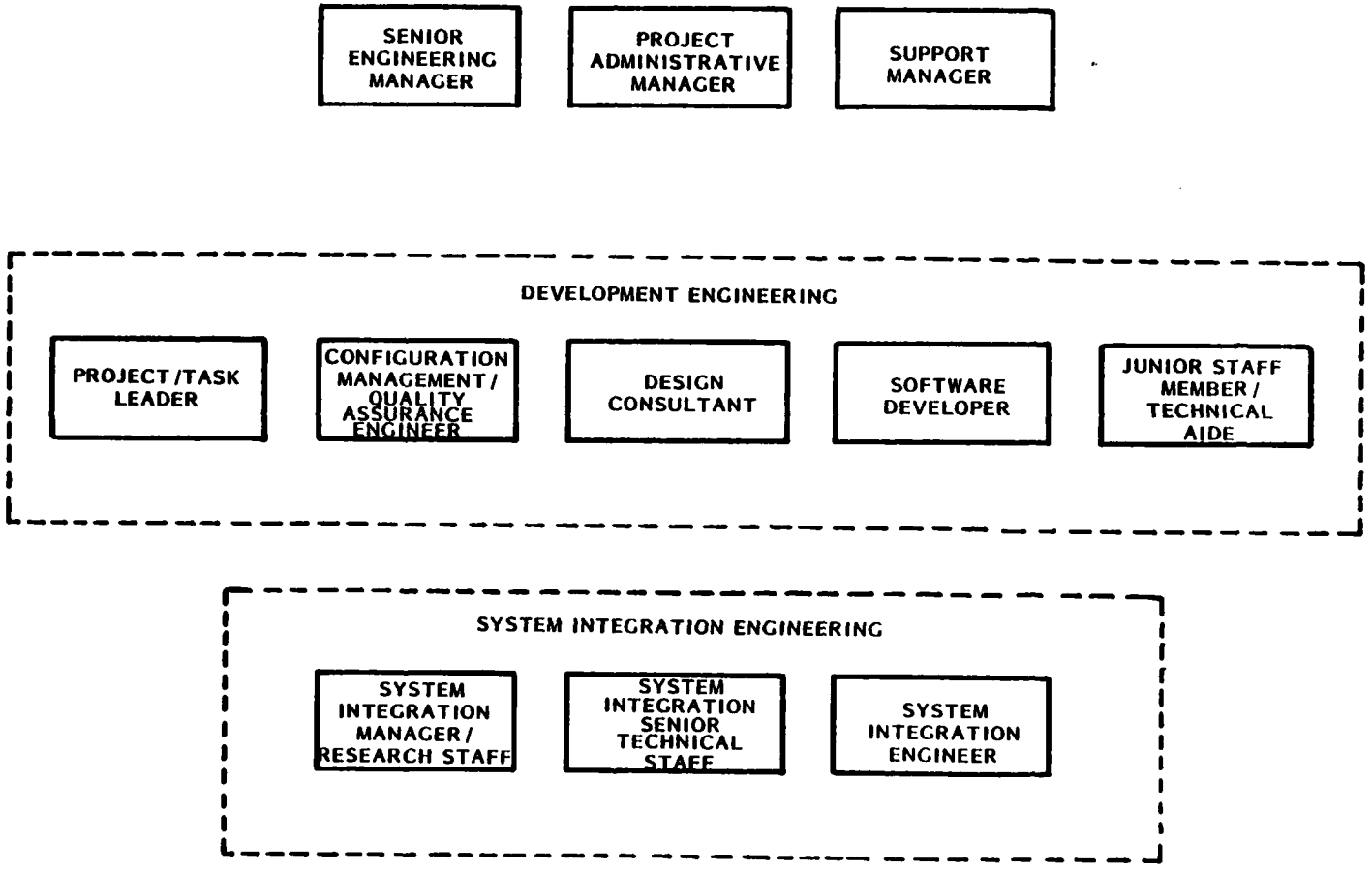


Figure 3-5. Secondary Breakdown of Development Engineering and System Integration Clusters

3.3.2 Proposed New Job Categories

Perhaps surprisingly, no further subclusters were found in the category labeled Software Developer. This is a somewhat disturbing result, since it indicates that today all software developers, regardless of experience, knowledge and abilities, tend to be involved in the same activities. Surely this is not the optimal utilization of manpower; on one hand there are people involved in activities for which they are not qualified; on the other hand there are senior designers who spend a large portion of their time in low-level activities.

The project team felt that Ada provides an opportunity for improving this situation. Since Ada has constructs directly related to high-level design, the qualifications needed for different implementation activities can be directly expressed in terms of Ada features that the individual must be able to use effectively.

It was speculated that the software development function must be further structured from the point of view of effective personnel utilization. New job categories were therefore proposed in this area (see Figure 3-6). The goal was to establish a clear separation of responsibility among individuals at different levels of expertise and knowledge. Once such a clear separation exists, one can safely provide each individual with the strictly minimal training necessary for his or her job function; the individual's technical responsibility will be - by definition - strictly commensurate to his or her knowledge and ability.

For example, at the beginning of the scale we place the category Programmer (as usual, the label is arbitrary, and does not necessarily match the use of the term in any particular organization). In defining the training needs for this individual, the goal was to establish the minimal knowledge necessary to contribute effectively to a project. For example, a Programmer should never use the low-level features of the language, which are potentially unsafe and therefore within the domain of more skilled designers.

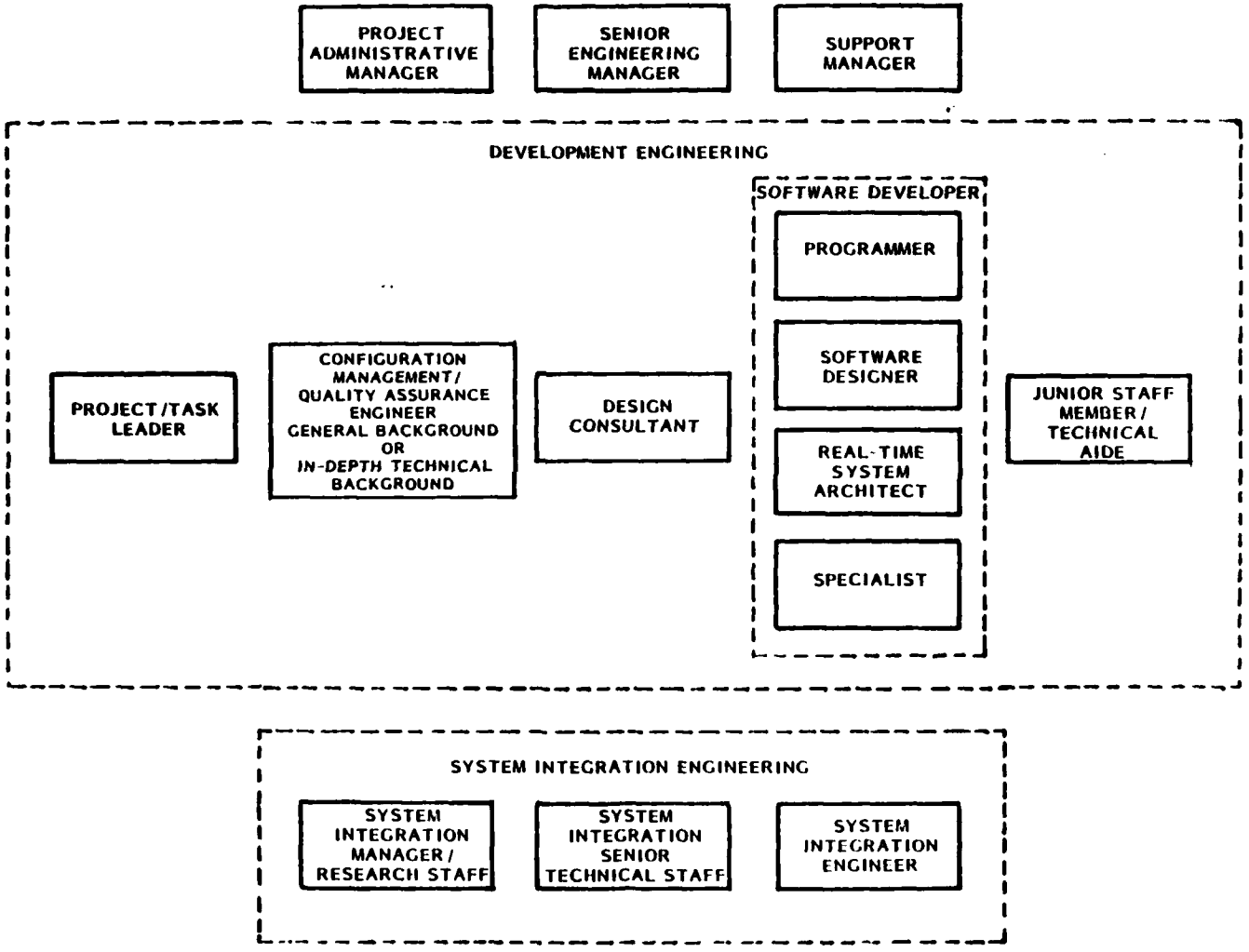


Figure 3-6. Final Job Classifications Including Breakdown of Software Developer Cluster

In essence, the objective of this approach is to:

a) improve the quality of software by clearly delineating the responsibility of people with different degrees of training and, at the same time,

b) minimize the amount of training required by the majority of the workforce.

3.3.3 Job Classification Categories

Fourteen job classifications resulted from the process described above. These are as follows:

- PROJECT ADMINISTRATIVE MANAGER: has little or no line management responsibility, but rather provides support in administrative areas (e.g., budgeting and scheduling).

- SENIOR ENGINEERING MANAGER: a manager of senior level technical people whose engineering decisions are not limited exclusively to software.

- SUPPORT MANAGER: The Support Manager is an individual with considerable experience who manages a support function which includes both technical and non-technical responsibilities.

- PROJECT/TASK LEADER: responsible for a project or a major task of a project. This person manages design and development, but does not actually perform these functions.

- CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER: does either configuration management and/or quality assurance. Some of these individuals are concerned with in-depth technical details while others concentrate on general issues.

- DESIGN CONSULTANT: in a staff function, this person has a very strong technical background and may serve as a resource for several projects at once or may be involved in research.

- PROGRAMMER: contributes at a basic level to software projects. This person operates under the guidance of more skilled individuals.

- SOFTWARE DESIGNER: participates at an advanced level in software design, but does not perform high-level real-time systems and concurrent software design.

- REAL-TIME SYSTEM ARCHITECT: responsible for high-level design of real-time systems and concurrent software. This person participates in all facets of embedded system design including hardware, system theory, and hardware/software partitioning, etc.

- SPECIALIST: a senior programmer who specializes in a particular area (e.g. numerical methods, graphics, database management, hardware diagnostics).

- JUNIOR STAFF MEMBER/TECHNICAL AIDE: assists programmers (e.g. data entry, submitting compilations, running tests, etc.)

- SYSTEM INTEGRATION MANAGER/RESEARCH STAFF: the person with broad technical experience whose responsibilities include management of system integration, hardware/software partitioning, etc.

- SYSTEM INTEGRATION SENIOR TECHNICAL STAFF: an individual who has significant technical responsibility, typically in the area of systems analysis. This person has only moderate management responsibility.

- SYSTEM INTEGRATION ENGINEER: principal activities involve design, coding, testing, etc; no management responsibility.

3.3.4 Issues Not Addressed

It is likely that the advent of Ada will have far-reaching repercussions on the work force structure. As programming evolves into software engineering, old job categories will become obsolete and new ones will appear; new career paths will likewise emerge, due to the different position of software in the system development and in the organization structure. Issues of such scale could clearly not be addressed by a project of this magnitude.

3.4 Generic Job Categories

As noted previously (see Section 3.2), SofTech was able to identify the following generic clusters through the statistical analysis of the Industry/Government Work Force Survey:

1. Project Administrative Manager
2. Senior Engineering Manager
3. Support Manager
4. Project/Task Leader
5. Configuration Management/Quality Assurance Engineer

6. Design Consultant

7. Software Developer

   - Programmer
   - Software Designer
   - Real-Time System Architect
   - Specialist

8. Junior Staff Member/Technical Aide

9. System Integration Manager/Research Staff

10. System Integration Senior Technical Staff

11. System Integration Engineer

Taken by themselves, these job titles are of little consequence. They represent what appeared to be reasonable descriptions of categories at the time this report was being prepared. More important however, is the data which constitute a given category. This data is presented in the subparagraphs below, which break out the following data for each job category:

1. Years of Software Development and/or Support (Table a)
2. Principal Outputs and Duties (Table b)
3. Programming Languages (Studied or Used) (Table c)
4. Knowledge of Methodologies (Table d)
5. Knowledge of Programming Constructs (Table e)
6. Knowledge of Programming Concepts (Table f)
7. Knowledge of Ada Programming Concepts (Table g)
8. Self-Described Titles (Table h)

The reader should bear in mind that "Software Developer" designates one category statistically, since additional meaningful clusters could not be derived from this category. As noted in Section 3.2.2 however, SofTech felt that additional functional specification was probably in order for an Ada oriented work force. Consequently, while the subcategories of Software Developer are identified, the background data for each is the same.

3.4.1 PROJECT ADMINISTRATIVE MANAGER

3.4.1.1 Job Summary

The Project Administrative Manager has little or no line management responsibility, but rather provides support in administrative areas (e.g, budgeting and scheduling).

3.4.1.2 Summary of Survey Results

The Project Administrative Manager typically is an individual with over ten years experience in software development or support, whose principal activities include technical management, quality assurance, and the formulation of policy and strategy. Structured programming and top-down design are the most frequently used methodologies for the group. FORTRAN and Assembler constitute the two most familiar programming languages for this category, and their general knowledge of programming concepts and constructs reflect this orientation. For instance, more people are familiar with fixed and floating types and conditional statements than with enumeration types, typed pointers and the importation of names. On the average, almost 50% of the group has some knowledge of Ada programming concepts.

TABLE 3-1b
PROJECT ADMINISTRATIVE MANAGER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank
updating training manuals	1.0
quality assurance	1.0
formulating policy	1.0
interviewing personnel	1.0
formulating strategy	1.0
technical advice to Configuration Control Board	1.5
management plans	1.5
technical management	1.5
contract negotiation	2.0
preparing budgets	2.0
technical management	2.0
interview sheets	2.0
preparing schedules	2.0
preparing management information reports	2.0
milestone charts	2.0
maintenance configuration procedures	2.0
library control	2.0
program management	2.0
hardware/software tradeoff evaluation	2.0

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was more with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-1c
PROJECT ADMINISTRATIVE MANAGER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	17
CMS-2	17
C	5
FORTRAN	46
COBOL	21
ASSEMBLER	44
PL/I	23
PASCAL	22
BASIC	31
ALGOL	15
RATFOR,WATFOR,WATFIV	9
MODULA	0
SIMULA	1
XPL	2
MMP	0
FORTH	2
Ada	14
LISP	7
SNOBOL	7
ECL	1
GPSS	12
SAS	2
PROTEGE	0
PPL	0
APL	12
Other	4

TABLE 3-1d
PROJECT ADMINISTRATIVE MANAGER
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	5	18	0	0
SADT	6	10	2	0
SREM	5	7	0	0
HIPO	5	14	15	9
Jackson Design	5	9	3	1
Structured Design	1	12	9	23
Warnier/Orr Design	6	9	4	0
N-S/Chapin Charts	7	9	3	2
Beamson Tables	3	1	0	0
Program Design Language	3	12	11	15
Structured Programming	0	5	8	33
Structured Walkthroughs	0	11	10	23
Top-Down Design	0	4	9	33
Top-Down Testing	1	5	11	28
Bottom-Up Design	0	11	22	12
Bachman Diagramming	1	3	0	0
Entity Diagrams	0	1	2	0
Data Abstraction	6	9	4	3
Other methodology	0	0	1	2

TABLE 3-1e
PROJECT ADMINISTRATIVE MANAGER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	2	11	11	6
floating point types	0	4	13	30
fixed point types	0	2	9	34
user defined types	2	8	13	18
pointers	0	5	13	28
typed pointers	1	15	8	15
ranges	0	13	10	22
records	1	8	10	25
variant records	3	13	5	14
object/type declarations	1	9	11	18
global variables	0	4	7	34
local variables	0	3	6	36
formal and actual parameters	0	1	12	31
reserved words	0	4	8	33
blocks	0	4	12	29
case statement	0	4	15	24
if/then/else statements	0	3	11	32
loop/for/while/until statements	0	3	11	32
exit statements (for loops)	0	6	14	26
procedures	0	3	12	31
functions	0	0	11	35
return statements	0	2	5	39
clusters/modules/packages	2	7	12	22
stubs	1	3	13	25
goto statements	0	3	15	26
comments	0	0	7	39
exception handlers	1	11	9	20
tasks/coroutines	2	11	8	19
other programming constructs	0	0	0	0

TABLE 3-1f
PROJECT ADMINISTRATIVE MANAGER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	2	8	4	4
data encapsulation (compools)	4	15	7	10
name scoping	3	10	2	9
name visibility	7	7	5	4
static and dynamic nesting	5	11	6	13
iteration	1	1	10	31
conditional statements	0	1	8	35
recursion	0	8	15	21
concurrency	1	8	11	16
strong typing	4	12	7	12
type conversion	3	6	16	17
data abstraction	4	14	6	10
generics	3	11	6	6
loop invariants	2	11	10	7
parameter binding	5	12	9	8
version numbers	1	5	11	21
other programming concepts	0	0	0	0

TABLE 3-1g
PROJECT ADMINISTRATIVE MANAGER
KNOWLEDGE OF ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	2	20	2	1
user-defined types	3	27	1	2
subtypes	6	19	0	1
derived types	8	12	1	2
real types	2	24	2	6
floating point types	0	23	4	9
fixed point types	0	22	6	7
record types	2	23	1	3
record types with discriminants	5	11	1	1
slices	3	7	1	1
aggregates	4	8	1	0
allocators	5	8	0	0
access types	4	15	1	0
overloading	4	10	0	1
packages	4	17	1	0
private types	1	15	0	1
scope	3	17	1	2
short circuiting	4	9	1	0
visibility	4	11	1	0
tasking	3	21	2	2
task types	4	17	2	1
rendezvous	2	13	3	1
entries	3	20	0	3
entry families	6	9	1	1
separate compilation	0	23	1	6
exceptions	3	18	2	1
generic program units	2	11	1	0
instantiation	4	11	0	0
elaboration	8	7	0	0
context specification	8	11	0	0
information hiding	1	18	0	2
mutual recursion	7	8	0	0
other Ada concepts	0	0	0	0

TABLE 3-1h
PROJECT ADMINISTRATIVE MANAGER
SELF-DESCRIBED TITLES

Chief, CAD/CAM Systems
Staff Specialist
Project Engineer
Software Design Specialist
Engineering Software Supervisor
Software Design Specialist
Software Engineering Specialist
Engineering Specialist
Engineering Specialist
Software Design Specialist
Software Engineering Specialist
Supervisor
Supervisor
Chief, Product Software
Software Design Specialist
Software Engineering Specialist
Senior Software Engineer
Engineering Chief
Supervisor
Software Development Manager
Senior System Analyst
Software Management
Software Manager
Engineering Specialist
R&D Engineer
Digital Signal Processing
Engineering Specialist
*
Senior Engineering Specialist
Engineering Specialist
Engineering Specialist
Consultant
Consultant
Consultant
Software Group Manager
Programmer Analyst
Unit Manager
Consultant
CAD Development Manager
Consultant
*
*
*
*
*
*

Key: * no title given on survey.

3.4.2 SENIOR ENGINEERING MANAGER

3.4.2.1 Job Summary

The Senior Engineering Manager is a manager of senior level technical people; his decisions are not limited exclusively to software.

3.4.2.2 Summary of Survey Results

The Senior Engineering Manager is typically a person with over ten years experience whose principal functions include technical management, hardware/software tradeoff evaluation, the review of system requirements and the tracking of cost and milestone data. The two most frequent languages listed as studied or used are FORTRAN and Assembler, though BASIC made a surprisingly strong showing. The most frequently used methodologies for this group are top-down design, top-down testing and structured programming. As a group, Senior Engineering Managers indicated most familiarity with programming constructs and concepts like conditional statements, functions, procedures, and global variables. They were much less familiar with enumeration types, typed pointers, stubs, variant records, tasks and coroutines, and data encapsulation. A significant number of people in this category (about one third) are familiar with Ada programming concepts such as packages, user defined types, private types, and tasking.

TABLE 3-2a
SENIOR ENGINEERING MANAGER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	0
TWO TO FIVE YEARS	1
FIVE TO TEN YEARS	3
OVER TEN YEARS	26
TOTAL	30

TABLE 3-2b
SENIOR ENGINEERING MANAGER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
cost data	1.0	conduct design review	2.0
milestone charts	1.0	conduct walkthroughs	2.0
analysis reports	1.0	formulation of policy	2.0
hardware/software tradeoff evaluation	1.0	support other	2.0
		program management	2.0
conduct requirements review	1.0	configuration management	2.0
		quality assurance	2.0
technical management	1.5	monitoring contracts	2.0
requirements specification	2.0	other development	2.0
interview sheets	2.0	support analysis	2.0
correspondence	2.0	support design	2.0
development other	2.0	conduct support design review	2.0
temporary Engineering Change Proposal	2.0	conduct support walkthrough	2.0
support test drivers	2.0	support technical management	2.0
technical advice to configuration control board	2.0	support policy formulation	2.0
		support program management	2.0
update MIL-STD spec	2.0	Software Configuration Control Board participation	2.0
library control	2.0		
maintaining configuration procedures	2.0	support configuration management	2.0
updating training manuals	2.0	support quality assurance	2.0
updating user manuals	2.0	support monitoring contracts	2.0
automated build systems	2.0		
management information reports	2.0	sales marketing	2.0
		preparing field engineering reports	2.0
version description documents	2.0		
version audits	2.0	prepare version audits	2.0
field engineering reports	2.0		
test drivers	2.0		
integration plans	2.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-2c
SENIOR ENGINEERING MANAGER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	8
CMS-2	13
C	4
FORTRAN	28
COBOL	15
ASSEMBLER	28
PL/I	14
PASCAL	19
BASIC	24
ALGOL	9
RATFOR,WATFOR,WATFIV	5
MODULA	1
SIMULA	2
XPL	1
MMP	0
FORTH	6
Ada	13
LISP	5
SNOBOL	8
ECL	0
GPSS	5
SAS	0
PROTEGE	0
PPL	0
APL	9
Other	12

TABLE 3-2d
SENIOR ENGINEERING MANAGER
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	2	9	1	0
SADT	1	5	1	1
SREM	2	5	0	0
HIPO	5	10	9	4
Jackson Design	5	2	1	1
Structured Design	1	5	9	13
Warnier/Orr Design	3	6	1	1
N-S/Chapin Charts	5	3	1	1
Beamson Tables	1	1	0	0
Program Design Language	3	5	13	6
Structured Programming	1	3	10	16
Structured Walkthroughs	2	9	9	9
Top-Down Design	0	2	10	19
Top-Down Testing	0	6	8	15
Bottom-Up Design	0	9	12	8
Bachman Diagramming	0	2	1	0
Entity Diagrams	3	2	2	0
Data Abstraction	1	5	6	4
Other methodology	0	0	2	0

TABLE 3-2e
SENIOR ENGINEERING MANAGER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	5	5	4
floating point types	1	5	6	18
fixed point types	1	3	6	19
user defined types	0	5	9	14
pointers	0	6	8	13
typed pointers	4	5	6	9
ranges	0	5	9	11
records	0	5	7	17
variant records	3	2	7	10
object/type declarations	1	2	7	15
global variables	0	3	6	20
local variables	0	3	6	19
formal and actual parameters	1	2	5	17
reserved words	1	3	7	16
blocks	0	5	8	12
case statement	1	4	8	17
if/then/else statements	0	1	7	23
loop/for/while/until statements	0	1	7	22
exit statements (for loops)	0	3	8	19
procedures	0	2	5	22
functions	0	3	5	21
return statements	0	2	5	23
clusters/modules/packages	2	7	6	12
stubs	0	9	11	7
goto statements	0	2	12	16
comments	0	1	10	19
exception handlers	1	5	8	12
tasks/coroutines	4	6	7	10
other programming constructs	0	0	1	1

TABLE 3-2f
SENIOR ENGINEERING MANAGER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Knowledge Level / Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	6	2	2	2
data encapsulation (compools)	7	6	5	4
name scoping	2	4	6	3
name visibility	5	3	4	4
static and dynamic nesting	5	4	6	5
iteration	1	3	7	18
conditional statements	1	0	6	22
recursion	1	5	12	10
concurrency	0	5	8	10
strong typing	3	6	6	6
type conversion	1	6	8	9
data abstraction	3	7	5	7
generics	5	6	3	5
loop invariants	2	4	6	5
parameter binding	2	9	2	6
version numbers	1	6	6	10
other programming concepts	0	0	0	0

TABLE 3-2g
SENIOR ENGINEERING MANAGER
KNOWLEDGE OF ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	3	6	4	2
user-defined types	2	11	1	4
subtypes	3	9	2	3
derived types	3	7	1	1
real types	1	8	5	5
floating point types	1	9	4	6
fixed point types	1	9	3	7
record types	0	11	3	5
record types with discriminants	4	9	0	0
slices	4	6	0	0
aggregates	3	5	0	1
allocators	2	7	2	0
access types	3	7	1	2
overloading	3	7	0	1
packages	3	10	1	1
private types	4	10	1	1
scope	2	12	2	0
short circuiting	2	4	1	0
visibility	1	9	1	1
tasking	3	10	0	4
task types	5	5	0	2
rendezvous	5	6	1	1
entries	4	7	2	1
entry families	6	3	1	1
separate compilation	3	5	2	6
exceptions	1	6	4	2
generic program units	8	3	1	1
instantiation	4	5	1	0
elaboration	4	3	0	1
context specification	4	4	0	0
information hiding	5	5	1	2
mutual recursion	5	4	1	1
other Ada concepts	0	0	0	0

TABLE 3-2h
SENIOR ENGINEERING MANAGER
SELF-DESCRIBED TITLES

*
Software Design Specialist
Senior Programmer
Software Design Specialist
Software Design Specialist
Principle Engineer
Project Manager
Data Processing Consultant
Senior System Analyst
Advanced R&D Engineer
*
R&D Engineer
Advanced R&D Engineer
Engineering Manager
Engineering Specialist
Software Engineer
Engineering Specialist
System Analyst
Consultant
Programmer
Consultant
Principle Programmer Analyst
Senior Consultant
Programmer Analyst
*
*
*
*
*
Computer Specialist

Key: * no title given on survey.

3.4.3 SUPPORT MANAGER

3.4.3.1 Job Summary

The Support Manager is an individual with considerable experience who manages a support function which includes both technical and non-technical responsibilities.

3.4.3.2 Summary of Survey Results

The Support Manager is typically an individual with over ten years experience in software development or support, whose principal outputs and duties range from contract negotiation, program management and configuration management to sales and marketing (see Table 3-3b). The broad responsibilities of the Support Manager suggest that he functions as a general resource within his organization. The two languages most familiar to this group are FORTRAN and Assembler; structured programming, top-down design, and top-down testing are the most familiar software engineering methodologies. As would be expected from the programming background, concepts and constructs such as iteration, conditional statements, functions and procedures are the most familiar, while enumeration and fixed point types are much less so. About half of the group indicated that they were aware of Ada concepts like user-defined types and separate compilation, but on the whole Ada concepts are relatively unknown.

TABLE 3-3a
SUPPORT MANAGER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	2
TWO TO FIVE YEARS	0
FIVE TO TEN YEARS	2
OVER TEN YEARS	10
TOTAL	14

TABLE 3-3b
SUPPORT MANAGER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
contract negotiation	1.0	configuration management	1.0
sales marketing	1.0	quality assurance	1.0
prepare version audits	1.0	monitoring contracts	1.0
library control	1.0	other development	1.0
correspondence	1.0	conduct support design review	1.0
development other	1.0	support policy formulation	1.0
other administrative tasks	1.0	conduct support walkthrough	1.0
temporary Engineering Change Proposals	1.0	attend support walkthrough	1.0
redlined documentation	1.0	support technical management	1.0
support test plans	1.0	support program management	1.0
support test drivers	1.0	Software Configuration Control Board	1.0
technical advice to Configuration Control Board	1.0	support configuration management	1.0
updated MIL-STD specification	1.0	support quality assurance	1.0
library control	1.0	support monitoring contracts	1.0
maintain configuration procedures	1.0	other support	1.0
updated training manuals	1.0	management plans	1.5
Software Trouble Reports	1.0	attend support design reviews	2.0
automated build systems	1.0	cost data	2.0
management information reports	1.0	formulating policy	2.0
version description documents	1.0	formulating strategy	2.0
version audits	1.0	interviewing personnel	2.0
field engineering reports	1.0	conduct requirements review	2.0
support other	1.0	Software Trouble Reports	2.0
formulation of policy	1.0	interview sheets	2.0
support other	1.0	analysis reports	2.0
formulation of policy	1.0		
formulation of strategy	1.0		
program management	1.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-3c
SUPPORT MANAGER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	5
CMS-2	8
C	0
FORTRAN	13
COBOL	5
ASSEMBLER	12
PL/I	6
PASCAL	3
BASIC	9
ALGOL	3
RATFOR,WATFOR,WATFIV	2
MODULA	0
SIMULA	0
XPL	0
MMP	0
FORTH	0
Ada	4
LISP	1
SNOBOL	2
ECL	0
GPSS	1
SAS	0
PROTEGE	0
PPL	0
APL	1
Other	5

TABLE 3-3d
SUPPORT MANAGER
KNOWLEDGE OF METHODOLOGIES

Knowledge Level / Methodology	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	3	1	0	0
SADT	0	4	2	1
SREM	2	1	0	0
HIPO	0	6	5	1
Jackson Design	3	2	0	1
Structured Design	0	4	5	4
Warnier/Orr Design	4	3	2	0
N-S/Chapin Charts	2	0	1	1
Beamson Tables	1	0	0	0
Program Design Language	3	3	3	2
Structured Programming	0	5	1	8
Structured Walkthroughs	0	4	3	5
Top-Down Design	0	5	1	8
Top-Down Testing	0	7	1	6
Bottom-Up Design	1	7	4	2
Bachman Diagramming	2	0	0	0
Entity Diagrams	1	2	0	0
Data Abstraction	3	5	0	1
Other methodology	0	0	0	0

TABLE 3-3e
SUPPORT MANAGER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Constructs \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	1	1	2	3
floating point types	0	1	6	7
fixed point types	0	2	6	6
user defined types	0	3	4	3
pointers	0	3	2	7
typed pointers	1	3	1	3
ranges	0	3	4	3
records	0	3	4	5
variant records	2	2	0	3
object/type declarations	1	1	2	5
global variables	0	1	6	6
local variables	0	2	6	5
formal and actual parameters	0	2	4	4
reserved words	0	2	3	7
blocks	1	2	3	3
case statement	0	1	4	7
if/then/else statements	0	2	3	8
loop/for/while/until statements	0	3	2	8
exit statements (for loops)	1	2	3	7
procedures	0	3	3	7
functions	0	4	3	7
return statements	0	3	2	9
clusters/modules/packages	0	4	2	8
stubs	1	3	3	5
goto statements	0	3	2	8
comments	0	1	3	9
exception handlers	1	1	5	5
tasks/coroutines	0	3	1	6
other programming constructs	0	0	0	0

TABLE 3-3f
SUPPORT MANAGER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	2	1	0	2
data encapsulation (compools)	0	4	1	4
name scoping	2	3	0	3
name visibility	2	2	0	2
static and dynamic nesting	1	2	2	4
iteration	0	6	2	6
conditional statements	0	4	3	6
recursion	2	4	2	5
concurrency	2	5	1	5
strong typing	3	1	1	4
type conversion	5	1	0	4
data abstraction	1	6	1	2
generics	2	5	1	2
loop invariants	3	1	0	5
parameter binding	4	0	0	3
version numbers	1	3	3	5
other programming concepts	0	0	1	0

TABLE 3-3g
SUPPORT MANAGER
KNOWLEDGE OF ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	1	4	1	0
user-defined types	2	8	0	1
subtypes	3	4	0	1
derived types	2	4	0	0
real types	2	4	0	2
floating point types	3	6	0	2
fixed point types	3	5	0	2
record types	5	2	0	2
record types with discriminants	3	2	0	0
slices	2	3	0	0
aggregates	2	2	0	1
allocators	1	3	0	0
access types	1	4	0	1
overloading	1	4	1	0
packages	4	5	0	0
private types	2	4	0	0
scope	2	3	0	2
short circuiting	2	1	0	1
visibility	1	3	0	1
tasking	4	5	0	2
task types	3	5	0	0
rendezvous	3	4	0	1
entries	5	2	0	2
entry families	4	2	0	0
separate compilation	2	7	0	2
exceptions	2	4	0	2
generic program units	2	4	1	0
instantiation	2	4	0	1
elaboration	2	2	0	0
context specification	2	2	0	0
information hiding	4	4	0	1
mutual recursion	3	2	0	0
other Ada concepts	0	0	0	0

TABLE 3-3h
SUPPORT MANAGER
SELF-DESCRIBED TITLES

Software Quality Assurance
Software Design Specialist
Software Design Specialist
Software Design Specialist
Project Engineer
*
Software Management
Manager
Line Manager
Manager
Software Development Manager
Manager
*
General Engineer

Key: * no title given on survey.

3.4.4 PROJECT/TASK LEADER

3.4.4.1 Job Summary

The Project/Task Leader is responsible for a project or a major task of a project. This person manages design and development, but does not actually perform these functions.

3.4.4.2 Summary of Survey Results

The Project/Task Leader is an individual with over ten years experience in software development or support whose principal outputs and duties involve preparing budgets and management information reports, program management and preparing technical reports. The group is most familiar with FORTRAN, Assembler, and BASIC. By far, the most frequently used methodologies are top-down design, top-down testing and bottom-up design. In general, the most familiar constructs and concepts are object/type declarations, global and local variables, formal and actual parameters, iteration, and conditional statements. Knowledge of Ada programming concepts for this group is relatively low (less than 10%).

TABLE 3-4a
PROJECT/TASK LEADER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	0
TWO TO FIVE YEARS	0
FIVE TO TEN YEARS	1
OVER TEN YEARS	5
TOTAL	6

TABLE 3-4b
PROJECT/TASK LEADER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank
preparing budgets	1.0
prepare management information reports	1.0
program management	1.0
prepare technical reports	1.0
formulating strategy	2.0
support program management	2.0
technical management	2.0
technical advice to Configuration Control Board	2.0
prepare system requirements documents	2.0
quality assurance	2.0
other development	2.5
preparing schedules	2.5
support monitor contracts	2.5
prepare field engineering reports	2.5
library control	2.5
preparing version audits	2.5
teaching	2.5
prepare redlined documents	2.5
formulation of policy	2.5
monitoring contracts	2.5
reviewing technical work of others	2.5
other support	2.5

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

2.5 indicates that this cluster as a group was tied with another cluster for the 2.0 rank.

TABLE 3-4c
PROJECT/TASK LEADER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	0
CMS-2	1
C	0
FORTRAN	6
COBOL	3
ASSEMBLER	5
PL/I	2
PASCAL	3
BASIC	4
ALGOL	1
RATFOR,WATFOR,WATFIV	1
MODULA	0
SIMULA	0
XPL	1
MMP	0
FORTH	0
Ada	2
LISP	0
SNOBOL	0
ECL	1
GPSS	0
SAS	0
PROTEGE	0
PPL	0
APL	1
Other	1

TABLE 3-4d
PROJECT/TASK LEADER
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	1	2	0	0
SADT	0	0	0	0
SREM	0	0	0	0
HIPO	1	2	2	0
Jackson Design	0	0	1	0
Structured Design	0	0	3	2
Warnier/Orr Design	1	0	0	0
N-S/Chapin Charts	0	0	0	0
Beamson Tables	0	0	0	0
Program Design Language	0	4	2	0
Structured Programming	0	0	4	2
Structured Walkthroughs	0	3	3	0
Top-Down Design	0	0	2	4
Top-Down Testing	0	2	0	4
Bottom-Up Design	0	3	0	3
Bachman Diagramming	0	0	0	1
Entity Diagrams	0	0	0	1
Data Abstraction	1	0	0	1
Other methodology	0	0	0	0

TABLE 3-4e
PROJECT/TASK LEADER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	1	0	0	2
floating point types	1	0	0	5
fixed point types	1	0	1	4
user defined types	0	1	1	3
pointers	0	1	1	4
typed pointers	1	2	1	2
ranges	0	0	1	4
records	1	0	1	4
variant records	0	1	0	1
object/type declarations	0	1	0	5
global variables	1	0	0	5
local variables	0	0	0	6
formal and actual parameters	0	0	0	6
reserved words	0	0	1	4
blocks	0	0	1	5
case statement	0	1	1	4
if/then/else statements	0	1	2	3
loop/for/while/until statements	0	0	2	3
exit statements (for loops)	0	0	2	3
procedures	0	1	2	3
functions	0	0	2	4
return statements	0	0	1	5
clusters/modules/packages	1	1	2	2
stubs	2	0	1	3
goto statements	0	1	1	4
comments	0	0	0	5
exception handlers	0	0	4	2
tasks/coroutines	1	0	3	2
other programming constructs	0	0	0	0

TABLE 3-4f
PROJECT/TASK LEADER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	1	0	1	1
data encapsulation (compools)	1	0	1	2
name scoping	1	0	1	1
name visibility	1	0	1	1
static and dynamic nesting	1	1	0	2
iteration	0	1	0	5
conditional statements	0	0	2	4
recursion	0	2	3	1
concurrency	1	1	2	0
strong typing	0	2	0	1
type conversion	0	3	1	1
data abstraction	1	1	1	1
generics	1	1	1	0
loop invariants	0	1	1	2
parameter binding	1	0	0	1
version numbers	0	1	3	2
other programming concepts	0	0	0	1

TABLE 3-4g
PROJECT/TASK LEADER
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	1	1	0
user-defined types	0	2	1	1
subtypes	0	1	1	1
derived types	0	2	1	0
real types	0	2	1	2
floating point types	0	2	1	2
fixed point types	0	3	1	1
record types	0	2	2	0
record types with discriminants	1	1	0	0
slices	1	1	0	0
aggregates	1	0	1	0
allocators	1	0	1	1
access types	1	0	2	0
overloading	1	0	0	1
packages	1	0	0	1
private types	0	1	1	0
scope	1	0	0	1
short circuiting	1	1	0	0
visibility	1	1	0	1
tasking	1	2	0	1
task types	1	2	0	1
rendezvous	0	3	0	1
entries	1	1	0	0
entry families	1	2	0	0
separate compilation	1	1	0	0
exceptions	1	2	1	0
generic program units	0	2	0	0
instantiation	1	2	0	0
elaboration	1	1	0	0
context specification	1	1	0	0
information hiding	1	0	1	0
mutual recursion	1	0	1	0
other Ada concepts	1	1	0	0

TABLE 3-4h
PROJECT/TASK LEADER
SELF-DESCRIBED TITLES

Electrical Engineer
Project Supervisor
Software Design Specialist
Principle Systems Analyst
Senior Applications Analyst
Consultant

3.4.5 CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER

3.4.5.1 Job Summary

The Configuration Management/Quality Assurance Engineer does either configuration management and/or quality assurance. Some of these individuals are concerned with in-depth technical details while others concentrate on general issues.

3.4.5.2 Summary of Survey Results

The Configuration Management/Quality Assurance Engineer category is typically an individual with over ten years experience in software development or support. (It should be noted here, that only five respondents comprise the Configuration Management/Quality Assurance Engineer category, and thus, generalizations regarding the data must be somewhat guarded.) The data indicate that the principal outputs and duties of this group include monitoring contracts, configuration management, quality assurance, program management, and preparing version audits. The group is most familiar with FORTRAN, Assembler, and COBOL. Structured programming, top-down design, top-down testing, and bottom-up design are the methodologies most widely used by respondents in this cluster. The group is most familiar with the following programming constructs and concepts: procedures, functions, conditional statements, and version numbers. In general the group's knowledge of Ada is weak.

TABLE 3-5a
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	0
TWO TO FIVE YEARS	0
FIVE TO TEN YEARS	1
OVER TEN YEARS	4
TOTAL	5

TABLE 3-5b
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
other support	1.0	formulation of policy	1.0
support monitoring contracts	1.0	support quality assurance	1.0
contract negotiation	1.0	hardware testing	1.5
support configuration management	1.0	prepare version description manuals	1.5
Software Configuration Control Board	1.0	formulation strategy	1.5
interviewing personnel	1.0	support technical management	1.5
support formulation policy	1.0	prepare temporary engineering report	1.5
prepare field engineering reports	1.0	formulation policy	1.5
other administrative tasks	1.0	formulating strategy	1.5
technical advice to Configuration Control Board	1.0	sales marketing	1.5
maintain configuration procedures	1.0	technical marketing	2.0
library control	1.0	prepare management information reports	2.0
prepare version audits	1.0	update MIL-STD specifications	2.0
reading technical publications	1.0	update training manuals	2.0
quality assurance	1.0	teaching	2.0
prepare redlined documentation	1.0	reviewing technical work	2.5
other development	1.0		
monitoring contracts	1.0		
support program management	1.0		
configuration management	1.0		
program management	1.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

2.5 indicates that this cluster as a group was tied with another cluster for the 2.0 rank.

TABLE 3-5c
ONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	1
CMS-2	1
C	0
FORTRAN	5
COBOL	4
ASSEMBLER	5
PL/I	3
PASCAL	1
BASIC	3
ALGOL	1
RATFOR,WATFOR,WATFIV	2
MODULA	0
SIMULA	0
XPL	0
MMP	0
FORTH	0
Ada	1
LISP	0
SNOBOL	0
ECL	0
GPSS	1
SAS	0
PROTEGE	0
PPL	0
APL	1
Other	1

TABLE 3-5d
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
KNOWLEDGE OF METHODOLOGIES

Knowledge Level / Methodology	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	0	1	0	0
SADT	0	0	0	0
SREM	0	1	0	0
HIPO	0	3	0	0
Jackson Design	1	0	0	0
Structured Design	0	1	3	1
Warnier/Orr Design	0	0	1	1
N-S/Chapin Charts	0	0	1	0
Beamson Tables	0	0	0	0
Program Design Language	1	1	0	1
Structured Programming	0	1	2	3
Structured Walkthroughs	0	1	4	1
Top-Down Design	0	1	3	2
Top-Down Testing	0	1	4	1
Bottom-Up Design	0	1	0	2
Bachman Diagramming	0	0	0	0
Entity Diagrams	0	0	0	0
Data Abstraction	1	0	1	0
Other methodology	0	1	0	0

TABLE 3-5e
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	1	2	0
floating point types	0	0	2	2
fixed point types	0	0	1	3
user defined types	0	1	1	2
pointers	0	0	0	2
typed pointers	1	0	1	1
ranges	0	0	2	2
records	0	0	2	3
variant records	0	0	2	2
object/type declarations	0	1	3	1
global variables	0	1	2	2
local variables	0	1	2	2
formal and actual parameters	0	1	1	2
reserved words	0	0	3	2
blocks	0	0	3	1
case statement	0	0	2	3
if/then/else statements	0	0	0	5
loop/for/while/until statements	0	1	1	3
exit statements (for loops)	0	1	0	4
procedures	0	0	0	5
functions	0	0	0	5
return statements	0	0	0	5
clusters/modules/packages	1	0	0	2
stubs	0	0	2	1
goto statements	0	0	0	5
comments	0	0	0	5
exception handlers	0	0	3	0
tasks/coroutines	0	0	4	0
other programming constructs	0	0	0	0

TABLE 3-5f
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	1	0	0	1
data encapsulation (compools)	0	0	3	0
name scoping	0	2	0	0
name visibility	0	1	1	0
static and dynamic nesting	0	0	2	1
iteration	0	0	0	2
conditional statements	0	0	0	3
recursion	0	0	0	2
concurrency	0	0	1	1
strong typing	0	[illegible]	1	1
type conversion	0	0	0	2
data abstraction	1	0	1	0
generics	1	1	0	0
loop invariants	0	0	0	2
parameter binding	0	1	1	0
version numbers	0	0	0	3
other programming concepts	0	0	0	0

TABLE 3-5g
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	1	1	0
user-defined types	0	1	1	0
subtypes	0	1	0	1
derived types	0	1	0	1
real types	0	1	0	1
floating point types	0	2	0	1
fixed point types	0	2	0	1
record types	0	2	0	1
record types with discriminants	0	1	1	0
slices	0	1	0	1
aggregates	0	0	1	0
allocators	0	0	1	0
access types	1	0	1	0
overloading	0	2	0	1
packages	0	1	1	1
private types	1	0	1	0
scope	0	2	0	0
short circuiting	0	0	1	0
visibility	1	1	0	0
tasking	1	1	1	0
task types	1	1	1	0
rendezvous	0	1	1	0
entries	0	2	1	0
entry families	0	2	0	0
separate compilation	0	2	0	1
exceptions	0	0	0	1
generic program units	1	1	0	0
instantiation	0	2	0	0
elaboration	0	1	1	0
context specification	1	0	1	0
information hiding	1	0	1	0
mutual recursion	0	0	1	0
other Ada concepts	0	0	0	0

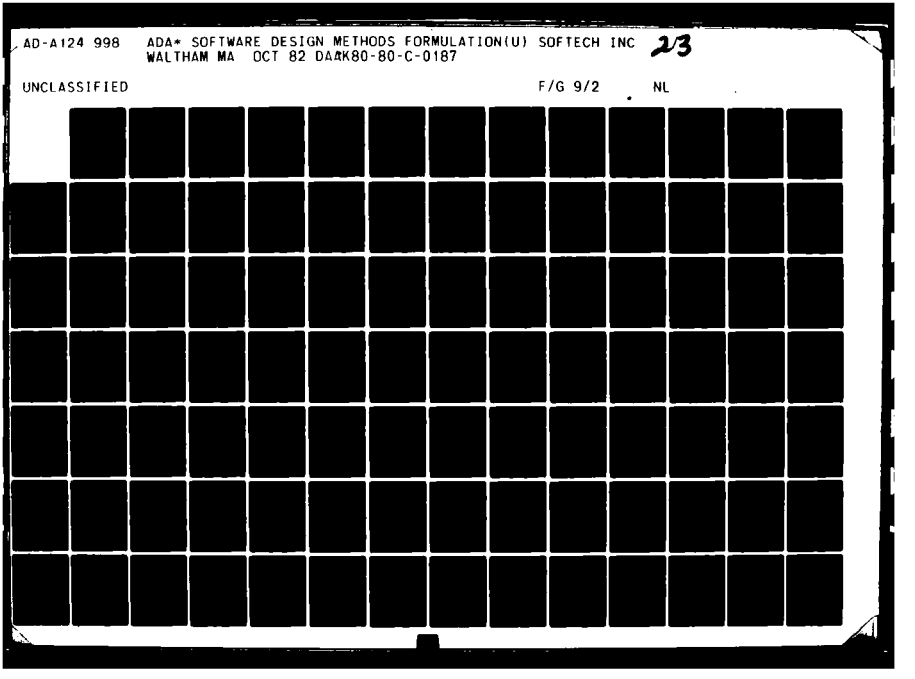
TABLE 3-5h
CONFIGURATION MANAGEMENT/QUALITY ASSURANCE ENGINEER
SELF-DESCRIBED TITLES

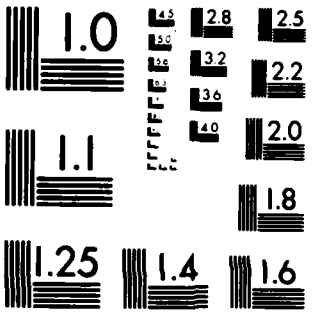
Computer Specialist
Computer Specialist
Computer Specialist
*
Computer Scientist

Key: * no title given on survey.

AD-A124 998 ADA* SOFTWARE DESIGN METHODS FORMULATION(U) SOFTECH INC 2/3
WALTHAM MA OCT 82 DAAK80-80-C-0187

UNCLASSIFIED F/G 9/2 NL





3.4.6 DESIGN CONSULTANT

3.4.6.1 Job Summary

The Design Consultant is in a staff function. This person has a very strong technical background and may serve as a resource for several projects at once or may be involved in research.

3.4.6.2 Summary of Survey Results

The Design Consultant is typically an individual with from two to five years experience in software development or support, whose principal outputs and duties include conducting and attending design reviews, coding and patching, teaching, and supporting and attending walkthroughs. The most frequent programming languages listed as studied or used were Assembler, FORTRAN, Pascal, Basic, and Ada. Structured programming and structured design are the most frequently used methodologies. As a group of the Design Consultants are most knowledgeable in the following programming constructs and concepts: functions, procedures, global and local variables, user defined types, and conditional statements. The Design Consultant category has a general familiarity with Ada concepts but very few members of this cluster have actually used Ada.

TABLE 3-6a
DESIGN CONSULTANT
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	2
TWO TO FIVE YEARS	7
FIVE TO TEN YEARS	3
OVER TEN YEARS	5
TOTAL	17

TABLE 3-6b
DESIGN CONSULTANT
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
support design	1.0	define module test cases	2.0
teaching	1.0	software module testing	2.0
conduct support design reviews	1.0	conduct walkthroughs	2.0
attend support design reviews	1.0	conduct requirements review	2.0
code/patch	1.0	attend requirements review	2.0
attend support walkthroughs	1.0	system analysis	2.0
conduct support walkthroughs	1.0	design	2.0
support technical management	1.5	conduct design reviews	2.0
formulating strategy	1.5	attend design reviews	2.0
other administrative tasks	2.0	code	2.0
attend walkthroughs	2.0	other development	2.5
support quality assurance	2.0	Software Configuration Control Board participation	2.5
update training manuals	2.0	formulation of policy	2.5
being trained	2.0	preparing budgets	2.5
functional system design	2.0	other support	2.5
functional module design	2.0	program management	2.5
define global data structures	2.0	quality assurance	2.5
define subsystems interface	2.0	monitoring contracts	2.5
define for own use	2.0	support monitoring contracts	2.5
coding	2.0	prepare version audits	2.5
debugging or modifying	2.0	library control	2.5
configuration management	2.0	preparing schedules	2.5
support analysis	2.0	prepare field engineering reports	2.5
system software testing	2.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

2.5 indicates that this cluster as a group was tied with another cluster for the 2.0 rank.

TABLE 3-6c
DESIGN CONSULTANT
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	3
CMS-2	6
C	1
FORTRAN	15
COBOL	8
ASSEMBLER	16
PL/I	6
PASCAL	13
BASIC	10
ALGOL	2
RATFOR,WATFOR,WATFIV	5
MODULA	0
SIMULA	1
XPL	0
MMP	0
FORTH	0
Ada	10
LISP	4
SNOBOL	4
ECL	0
GPSS	2
SAS	0
PROTEGE	0
PPL	0
APL	4
Other	3

TABLE 3-6d
DESIGN CONSULTANT
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	1	5	2	0
SADT	0	3	0	0
SREM	0	2	0	0
HIPO	3	8	1	1
Jackson Design	1	3	0	0
Structured Design	0	4	8	5
Warnier/Orr Design	0	1	2	0
N-S/Chapin Charts	2	1	1	0
Beamson Tables	0	0	0	0
Program Design Language	1	3	8	5
Structured Programming	0	1	8	8
Structured Walkthroughs	0	4	8	5
Top-Down Design	0	1	8	8
Top-Down Testing	0	4	8	5
Bottom-Up Design	0	8	7	0
Bachman Diagramming	1	1	0	0
Entity Diagrams	0	2	2	0
Data Abstraction	3	2	5	0
Other methodology	0	0	2	0

TABLE 3-6e
DESIGN CONSULTANT
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Constructs \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	1	3	7	1
floating point types	0	6	6	5
fixed point types	0	4	5	8
user defined types	1	3	9	3
pointers	0	3	7	6
typed pointers	3	2	5	5
ranges	0	4	9	3
records	0	4	7	6
variant records	2	3	6	3
object/type declarations	0	0	10	4
global variables	0	1	5	11
local variables	0	1	5	11
formal and actual parameters	0	2	5	8
reserved words	0	2	5	9
blocks	0	3	4	9
case statement	0	1	6	9
if/then/else statements	0	0	7	9
loop/for/while/until statements	0	0	7	9
exit statements (for loops)	0	2	9	5
procedures	0	0	5	11
functions	0	0	7	9
return statements	0	0	5	11
clusters/modules/packages	0	3	5	7
stubs	1	6	6	3
goto statements	0	4	8	4
comments	0	0	6	10
exception handlers	0	7	5	3
tasks/coroutines	0	8	4	3
other programming constructs	0	1	1	0

TABLE 3-6f
DESIGN CONSULTANT
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	0	4	4	0
data encapsulation (compools)	2	3	4	2
name scoping	2	3	3	4
name visibility	1	4	3	2
static and dynamic nesting	3	4	6	1
iteration	0	4	7	6
conditional statements	0	1	5	11
recursion	0	7	7	3
concurrency	1	10	4	0
strong typing	3	6	3	2
type conversion	1	4	6	2
data abstraction	3	5	4	1
generics	0	7	3	0
loop invariants	3	4	3	1
parameter binding	3	3	4	3
version numbers	1	4	7	3
other programming concepts	0	0	0	0

TABLE 3-6g
DESIGN CONSULTANT
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	2	8	1	1
user-defined types	2	10	1	1
subtypes	3	9	1	0
derived types	4	8	0	0
real types	3	10	0	1
floating point types	2	11	0	1
fixed point types	2	11	0	1
record types	4	8	1	1
record types with discriminants	3	8	0	0
slices	3	6	0	0
aggregates	3	6	0	0
allocators	4	5	0	0
access types	2	6	1	0
overloading	6	7	0	0
packages	3	8	1	0
private types	4	7	1	0
scope	2	8	0	1
short circuiting	2	5	0	0
visibility	3	6	0	1
tasking	5	8	1	0
task types	2	7	1	0
rendezvous	2	8	0	0
entries	2	7	0	0
entry families	2	3	0	0
separate compilation	4	11	0	0
exceptions	4	8	1	0
generic program units	4	5	1	0
instantiation	4	7	1	0
elaboration	3	3	0	0
context specification	3	4	0	0
information hiding	2	10	0	0
mutual recursion	5	5	0	0
other Ada concepts	0	0	0	0

TABLE 3-6h
DESIGN CONSULTANT
SELF-DESCRIBED TITLES

R&D Engineer
Senior Software Engineer
Senior Analyst
*
*
Programmer
*
Senior Software Engineer
Software Engineer
*
Senior System Analyst
Software R&D Engineer
Senior Engineer
Supervising System Analyst
Software Design Specialist
*
Senior Programmer Analyst

Key: * no title given on survey.

3.4.7 SOFTWARE DEVELOPER

3.4.7.1 Job Summaries

PROGRAMMER

The Programmer contributes at a basic level to software projects. This person operates under the guidance of more skilled individuals.

SOFTWARE DESIGNER

The Software Designer participates at an advanced level in software design, but does not perform high-level real-time systems and concurrent software design.

REAL-TIME SYSTEM ARCHITECT

The Real-Time System Architect is responsible for high-level design of real-time systems and concurrent software. This person participates in all facets of embedded system design including hardware, system theory, and hardware/software partitioning, etc.

SPECIALIST

The Specialist is a senior programmer who specializes in a particular area (e.g. numerical methods, graphics, database management, hardware diagnostics).

3.4.7.2 Summary of Survey Results

The Software Developer cluster comprises four job categories - Programmer, Software Designer, Real-Time System Architect, and Specialist. As discussed in Section 3.3.2, these job categories were arrived at by SofTech and not by a further statistical breakdown of the Software Developer cluster; they are thought to represent a structuring of job function which will result from the introduction of Ada into the software development process. The data presented in this section is therefore, for the entire Software Developer cluster and not for the individual categories within the cluster.

The respondents in the Software Developer category have a wide range of experience in software development or support. Forty-one percent have over ten years experience, twenty-two percent have five to ten years experience, twenty-seven percent have two to five years experience, and seven percent have less than two years experience. The principal outputs and duties of this cluster reflect an equally wide range, from technical management to coding. By far, FORTRAN and Assembler are the most familiar programming languages for the group, though half indicated a knowledge of Pascal and PL/I and sixty-four percent know BASIC. The most popular methodologies for the Software Developer group are structured programming, top-down design, and structured design. The programming concepts most familiar to the group are iteration, conditional statements, functions, procedures, and global and local variables. There appears to be moderate familiarity with Ada programming concepts.

TABLE 3-7a
SOFTWARE DEVELOPER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	10
TWO TO FIVE YEARS	36
FIVE TO TEN YEARS	29
OVER TEN YEARS	54
TOTAL	129

TABLE 3-7b
SOFTWARE DEVELOPER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
technical management	1.0	ocumenting test results	1.0
preparing schedules	1.0	prep trouble reports	1.0
reviewing technical work	1.0	analyze trouble reports	1.0
functional system design	1.0	conduct requirements review	1.0
functional module design	1.0	attend requirements review	1.0
define global data structures	1.0	attend walkthroughs	1.0
define subsystem interfaces	1.0	esign	1.0
define data structures and and algorithms for own use	1.0	conduct design review	1.0
		attend design review	1.0
		code	1.0
		conduct walkthroughs	1.0
coding	1.0	technical management	1.0
debugging or modifying	1.0	support analysis	1.0
preparing system report documents	1.0	support design	2.0
		prepare redlined documentation	2.5
system analysis	1.0	code patch	2.5
prepare user manuals	1.0	Software Configuration Control Board participation	2.5
documenting code	1.0		
defining test cases	1.0		
prepare test drivers	1.0		
prepare test plans	1.0		
system software test	1.0		
defining module test cases	1.0		
software module test	1.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

2.5 indicates that this cluster as a group was tied with another cluster for the 2.0 rank.

TABLE 3-7c
SOFTWARE DEVELOPER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	25
CMS-2	44
C	15
FORTRAN	123
COBOL	68
ASSEMBLER	122
PL/I	67
PASCAL	64
BASIC	82
ALGOL	29
RATFOR,WATFOR,WATFIV	33
MODULA	3
SIMULA	3
XPL	4
MMP	0
FORTH	8
Ada	28
LISP	26
SNOBOL	25
ECL	0
GPSS	15
SAS	1
PROTEGE	0
PPL	0
APL	32
Other	36

TABLE 3-7d
SOFTWARE DEVELOPER
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	16	20	7	0
SADT	5	3	7	0
SREM	3	8	1	1
HIPO	16	44	26	9
Jackson Design	11	10	5	6
Structured Design	5	13	34	71
Warnier/Orr Design	13	14	2	3
N-S/Chapin Charts	13	12	6	1
Beamson Tables	3	1	1	0
Program Design Language	5	32	25	51
Structured Programming	0	9	30	90
Structured Walkthroughs	4	36	42	37
Top-Down Design	1	8	32	86
Top-Down Testing	4	28	37	55
Bottom-Up Design	6	53	39	19
Bachman Diagramming	6	7	1	1
Entity Diagrams	7	6	0	1
Data Abstraction	14	34	15	10
Other methodology	0	0	0	3

TABLE 3-7e
SOFTWARE DEVELOPER
PROGRAMMING CONSTRUCTS

Constructs \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	9	26	17	18
floating point types	3	22	44	61
fixed point types	2	11	27	83
user defined types	6	28	34	43
pointers	2	16	31	74
typed pointers	10	26	20	34
ranges	5	27	28	49
records	2	19	34	70
variant records	12	29	21	28
object/type declarations	5	27	29	47
global variables	2	7	23	96
local variables	2	9	20	97
formal and actual parameters	7	7	25	70
reserved words	3	17	25	82
blocks	1	17	29	70
case statement	4	12	30	77
if/then/else statements	1	7	23	98
loop/for/while/until statements	1	9	26	93
exit statements (for loops)	1	27	35	64
procedures	2	8	25	94
functions	1	10	31	87
return statements	2	9	21	95
clusters/modules/packages	6	25	32	48
stubs	3	19	33	57
goto statements	0	22	36	70
comments	0	6	17	105
exception handlers	6	33	29	47
tasks/coroutines	9	36	27	41
other programming constructs	0	0	0	0

TABLE 3-7f
SOFTWARE DEVELOPER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	12	11	5	9
data encapsulation (compools)	17	26	21	20
name scoping	14	19	9	25
name visibility	14	23	6	21
static and dynamic nesting	22	24	15	23
iteration	3	14	33	75
conditional statements	3	5	25	95
recursion	1	35	45	37
concurrency	7	42	28	27
strong typing	9	29	17	19
type conversion	9	27	30	36
data abstraction	20	36	15	21
generics	15	28	15	11
loop invariants	19	23	16	13
parameter binding	19	26	14	11
version numbers	8	20	27	45
other programming concepts	0	0	0	2

TABLE 3-7g
SOFTWARE DEVELOPER
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	15	33	7	7
user-defined types	14	45	8	12
subtypes	20	31	7	5
derived types	23	22	7	3
real types	9	47	13	17
floating point types	7	51	13	20
fixed point types	5	50	9	24
record types	12	41	11	17
record types with discriminants	20	24	3	4
slices	13	15	6	4
aggregates	15	16	2	3
allocators	14	19	2	3
access types	15	27	3	6
overloading	12	24	3	3
packages	10	37	6	4
private types	10	33	5	2
scope	9	37	4	6
short circuiting	10	17	2	1
visibility	13	33	6	3
tasking	17	40	5	11
task types	18	31	3	5
rendezvous	5	31	2	2
entries	8	32	3	9
entry families	14	12	2	1
separate compilation	7	49	6	16
exceptions	10	39	6	10
generic program units	12	26	5	2
instantiation	8	19	1	3
elaboration	9	18	2	2
context specification	14	23	1	3
information hiding	10	38	3	4
mutual recursion	12	22	2	2
other Ada concepts	0	0	0	0

TABLE 3-7h
SOFTWARE DEVELOPER
SELF-DESCRIBED TITLES

Senior Scientific Programmer	Engineer	*
Scientific Programmer	Engineering Specialist	*
Senior Software Engineer	R&D Engineer	Diagnostic Software
Principal Scientific Programmer	Software Design Specialist	*
Principle Scientific Programmer	*	*
Software Engineer	Software Design Specialist	*
Software Design Specialist	Program Analyst	*
Computer Specialist	*	*
Engineer	Consultant	*
Senior Engineer	Software Design Specialist	*
Software Engineer	*	*
System Engineering Specialist	Senior Software Engineer	*
Junior Programmer	Senior Programmer Analyst	*
Digital Signal Processing	Engineering Software Supervisor	*
*	Programmer Analyst	*
R&D Engineer	Senior GS Analyst	*
Engineer	Principle Programmer Analyst	*
*	Analyst	Research Engineer
Junior Programmer	Government Program Analyst	*
Advanced Development Engineer	Government Program Analyst	*
Engineering Specialist	Principle Programmer Analyst	*
*	Team Leader	*
R&D Engineer	Computer Specialist	Principle Scientific Programmer
Programmer Analyst	*	*
Programmer	Programmer	*
R&D Engineer	Principle Programmer Analyst	*
Senior Engineer	Firmware Design Engineer	*
Software Engineer	System Analyst	*
Engineer	Consultant	*
Software R&D Engineer	Principle Engineer	*
Communications Software	Senior Programmer Analyst	Electrical Engineer
Advanced R&D Engineer	Software Engineer	*
Supervisor	Programmer	Senior Engineering Specialist
Senior Software Engineer	Programmer Analyst	Project Leader
R&D Engineer	System Design Specialist	Senior Software Engineer
Research Engineer	Associate Programmer Analyst	Senior Engineer
Software Engineer	Programmer	Electronic Technician
Engineering Specialist	Programmer Analyst	Electrical Engineer
Data Processing Consultant	Senior Engineer	Computer Specialist
R&D Engineer	Quality Assurance Engineer	*
Senior Engineer	*	Software Engineer
Programmer	*	Electrical Engineer
Senior Programmer	*	Electrical Engineer

Key:: * no title given on survey.

3.4.8 JUNIOR STAFF MEMBER/TECHNICAL AIDE

3.4.8.1 Job Summary

The Junior Staff Member/Technical Aide assists programmers (e.g. data entry, submitting compilations, running tests, etc.)

3.4.8.2 Summary of Survey Results

The Junior Staff Member/Technical Aide is typically an individual with less than five or over ten years of experience in software development or support. His principal outputs and duties include being trained, updating MIL STD specifications, and preparing temporary engineering reports and version description manuals. The group is most familiar with FORTRAN, Assembler, COBOL, and Basic. In knowledge of methodologies, structured design, structured programming, and structured walkthroughs predominate. The most familiar programming constructs and concepts for the group are records, global and local variables, blocks, case statements, iteration, and conditional statements. The group is marginally familiar with Ada concepts.

TABLE 3-8a
JUNIOR STAFF MEMBER/TECHNICAL AIDE
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	1
TWO TO FIVE YEARS	4
FIVE TO TEN YEARS	0
OVER TEN YEARS	4
TOTAL	9

TABLE 3-8b
JUNIOR STAFF MEMBER/TECHNICAL AIDE
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank
being trained	1.0
update MIL STD specs	1.0
preparing temporary engineering reports	1.5
sales marketing	1.5
hardware testing	1.5
prepare version description manuals	1.5
formulating policy	1.5
prepare trouble reports	2.0
reading technical publications	2.0
updating training manuals	2.0
analysis trouble reports	2.0
attend support design review	2.0
preparing user manuals	2.0
documentation test results	2.0
conduct support design review	2.0
attend support walkthroughs	2.0
documenting code	2.0
defining test cases	2.0
preparing test plans	2.0
preparing budgets	2.5
code/patch	2.5

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

2.5 indicates that this cluster as a group tied with another cluster for the 2.0 rank.

TABLE 3-8c
JUNIOR STAFF MEMBER/TECHNICAL AIDE
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	2
CMS-2	2
C	0
FORTRAN	8
COBOL	6
ASSEMBLER	7
PL/I	1
PASCAL	5
BASIC	7
ALGOL	3
RATFOR,WATFOR,WATFIV	2
MODULA	0
SIMULA	0
XPL	0
MMP	0
FORTH	1
Ada	1
LISP	2
SNOBOL	2
ECL	0
GPSS	1
SAS	0
PROTEGE	0
PPL	0
APL	4
Other	2

TABLE 3-8d
JUNIOR STAFF MEMBER/TECHNICAL AIDE
KNOWLEDGE OF METHODOLOGIES

Knowledge Level / Methodology	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	0	0	2	0
SADT	0	1	1	0
SREM	0	1	0	0
HIPO	0	1	4	0
Jackson Design	0	1	1	0
Structured Design	0	0	0	8
Warnier/Orr Design	0	1	1	0
N-S/Chapin Charts	3	2	0	0
Beamson Tables	0	1	0	0
Program Design Language	0	1	2	6
Structured Programming	0	0	0	9
Structured Walkthroughs	0	1	0	8
Top-Down Design	0	2	1	6
Top-Down Testing	0	1	3	4
Bottom-Up Design	0	4	1	3
Bachman Diagramming	1	2	0	0
Entity Diagrams	0	1	0	0
Data Abstraction	0	1	0	1
Other methodology	0	0	0	1

TABLE 3-8e
JUNIOR STAFF MEMBER/TECHNICAL AIDE
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	2	0	2
floating point types	0	2	1	6
fixed point types	0	1	2	6
user defined types	0	1	1	6
pointers	0	0	0	9
typed pointers	0	1	2	4
ranges	0	1	1	5
records	0	1	0	8
variant records	0	1	2	4
object/type declarations	0	1	3	3
global variables	0	0	0	9
local variables	0	0	0	9
formal and actual parameters	0	0	0	8
reserved words	0	0	1	8
blocks	0	0	0	9
case statement	0	0	1	8
if/then/else statements	0	0	1	8
loop/for/while/until statements	0	0	1	8
exit statements (for loops)	0	2	1	5
procedures	0	0	1	7
functions	0	0	1	8
return statements	0	1	0	8
clusters/modules/packages	0	1	0	7
stubs	0	0	1	7
goto statements	0	1	1	7
comments	0	0	0	9
exception handlers	0	1	0	7
tasks/coroutines	0	2	2	5
other programming constructs	0	0	0	0

TABLE 3-8f
JUNIOR STAFF MEMBER/TECHNICAL AIDE
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	0	1	0	2
data encapsulation (compools)	0	2	1	2
name scoping	1	1	0	2
name visibility	1	1	1	2
static and dynamic nesting	1	1	3	3
iteration	0	0	2	6
conditional statements	0	0	1	8
recursion	0	0	5	3
concurrency	0	2	3	2
strong typing	0	1	1	3
type conversion	0	2	1	3
data abstraction	1	2	1	2
generics	0	2	0	2
loop invariants	2	1	0	3
parameter binding	0	1	2	1
version numbers	0	0	0	5
other programming concepts	0	0	0	0

TABLE 3-8g
JUNIOR STAFF MEMBER/TECHNICAL AIDE
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	0	2	0	1
user-defined types	0	3	0	2
subtypes	0	3	0	2
derived types	0	3	0	1
real types	0	3	1	1
floating point types	0	4	0	1
fixed point types	0	4	1	1
record types	0	3	0	2
record types with discriminants	0	2	0	1
slices	1	1	0	1
aggregates	1	1	0	2
allocators	0	0	1	1
access types	0	1	0	1
overloading	0	2	0	1
packages	1	2	0	1
private types	0	3	0	1
scope	0	2	0	2
short circuiting	1	1	0	1
visibility	1	1	0	1
tasking	0	3	1	1
task types	0	2	0	1
rendezvous	0	2	0	1
entries	0	2	1	2
entry families	0	1	0	1
separate compilation	0	3	1	1
exceptions	0	3	0	1
generic program units	2	2	0	1
instantiation	1	0	0	1
elaboration	1	0	0	1
context specification	0	1	0	1
information hiding	1	1	0	1
mutual recursion	0	1	0	1
other Ada concepts	0	0	0	0

TABLE 3-8h
JUNIOR STAFF MEMBER/TECHNICAL AIDE
SELF-DESCRIBED TITLES

Consultant
*
Senior Engineer
Staff Consultant
R&D Engineer
Programmer Analyst
*
Programmer
Senior Engineer

Key: * no title given on survey.

3.4.9 SYSTEM INTEGRATION MANAGER/RESEARCH STAFF

3.4.9.1 Job Summary

The System Integration Manager/Research Staff is a person with broad technical experience whose responsibilities include management of system integration, hardware/software partitioning, etc.

3.4.9.2 Summary of Survey Results

The System Integration Manager/Research Staff is typically a person with over ten years experience in software development or support. His principal outputs and duties include program management, technical management, conducting requirements reviews, hardware testing, and formulating policy and strategy. FORTRAN, Basic, and Assembler are the programming languages most frequently studied or used by the group. The following programming constructs and concepts are most familiar: floating and fixed point types, functions, procedures, iteration, and conditional statements. The group has marginal familiarity with Ada programming concepts.

TABLE 3-9a
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	2
TWO TO FIVE YEARS	2
FIVE TO TEN YEARS	2
OVER TEN YEARS	7
TOTAL	13

TABLE 3-9b
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
program management	1.0	prepare redlined documentation	1.5
conduct requirements reviews	1.0	support monitoring contracts	1.5
contract negotiation	1.0	support quality assurance	1.5
reviewing technical work	1.0	Software Configuration Control Board participation	1.5
reading technical publications	1.0	preparing budgets	1.5
support program management	1.0	preparing management information reports	1.5
technical management	1.0	quality assurance	1.5
interviewing personnel	1.0	defining global data structures	2.0
preparing schedules	1.0	attending design reviews	2.0
support formulation policy	1.0	preparing system requirements documentation	2.0
prepare field engineering report	1.0	quality assurance	2.0
support technical management	1.0	technical advice to Configuration Control Board	2.0
monitoring contracts	1.0	other administrative tasks	2.0
update MIL-STD specification	1.0	preparing temporary engineering reports	2.0
library control	1.0	preparing technical reports	2.0
prepare version audits	1.0	attend walkthroughs	2.0
formulation of policy	1.0	support configuration management	2.0
attend requirements reviews	1.0	conduct design reviews	2.0
hardware testing	1.0	updating training manuals	2.0
formulating strategy	1.5	other support	2.0
formulating policy	1.5		
sales marketing	1.5		
maintain configuration procedures	1.5		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-9c
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	1
CMS-2	2
C	2
FORTRAN	11
COBOL	6
ASSEMBLER	7
PL/I	4
PASCAL	4
BASIC	8
ALGOL	2
RATFOR,WATFOR,WATFIV	2
MODULA	0
SIMULA	0
XPL	0
MMP	0
FORTH	1
Ada	3
LISP	1
SNOBOL	1
ECL	0
GPSS	2
SAS	0
PROTEGE	0
PPL	0
APL	1
Other	6

TABLE 3-9d
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	1	3	0	0
SADT	1	2	0	0
SREM	0	1	1	0
HIPO	1	2	2	0
Jackson Design	0	1	0	0
Structured Design	0	6	4	2
Warnier/Orr Design	0	1	0	0
N-S/Chapin Charts	0	1	0	0
Beamson Tables	0	0	0	0
Program Design Language	2	5	2	1
Structured Programming	1	5	3	3
Structured Walkthroughs	0	6	2	1
Top-Down Design	1	5	3	3
Top-Down Testing	1	6	2	3
Bottom-Up Design	1	8	1	1
Bachman Diagramming	0	1	0	0
Entity Diagrams	0	2	0	0
Data Abstraction	0	2	2	0
Other methodology	0	1	0	0

TABLE 3-9e
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Knowledge Level / Constructs	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	1	2	2	3
floating point types	0	4	5	2
fixed point types	0	3	5	3
user defined types	1	3	4	2
pointers	0	4	2	4
typed pointers	1	5	2	1
ranges	0	7	2	1
records	1	5	2	2
variant records	1	3	3	1
object/type declarations	1	1	4	1
global variables	2	4	3	2
local variables	1	3	4	3
formal and actual parameters	1	3	2	3
reserved words	0	5	2	3
blocks	1	5	2	2
case statement	1	4	2	2
if/then/else statements	1	4	3	3
loop/for/while/until statements	1	4	3	3
exit statements (for loops)	1	4	3	2
procedures	2	2	4	3
functions	2	2	4	3
return statements	1	3	3	4
clusters/modules/packages	2	1	5	2
stubs	2	4	3	1
goto statements	1	1	4	5
comments	1	1	4	5
exception handlers	2	4	2	0
tasks/coroutines	1	2	2	2
other programming constructs	0	0	0	0

TABLE 3-9f
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	1	0	0	0
data encapsulation (compools)	1	5	1	1
name scoping	1	1	2	1
name visibility	3	0	1	1
static and dynamic nesting	1	3	1	3
iteration	1	1	5	4
conditional statements	1	1	5	4
recursion	0	3	4	1
concurrency	0	2	1	3
strong typing	1	3	1	0
type conversion	0	3	1	2
data abstraction	0	4	0	1
generics	0	2	1	1
loop invariants	1	0	0	2
parameter binding	1	0	0	1
version numbers	1	3	2	1
other programming concepts	0	0	0	0

TABLE 3-9g
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	2	4	1	0
user-defined types	2	4	1	0
subtypes	2	3	1	0
derived types	2	3	1	0
real types	1	5	0	0
floating point types	1	5	1	0
fixed point types	1	5	1	0
record types	2	4	1	0
record types with discriminants	3	2	1	0
slices	4	0	0	0
aggregates	3	2	0	0
allocators	3	1	0	0
access types	1	5	0	0
overloading	2	3	0	0
packages	2	3	1	0
private types	0	5	0	0
scope	2	3	1	0
short circuiting	2	0	0	0
visibility	2	3	1	0
tasking	4	2	0	0
task types	4	2	0	0
rendezvous	2	4	0	0
entries	2	3	0	0
entry families	2	1	0	0
separate compilation	2	3	1	0
exceptions	3	3	0	0
generic program units	3	3	0	0
instantiation	3	2	1	0
elaboration	2	1	0	0
context specification	3	1	0	0
information hiding	3	2	0	0
mutual recursion	3	1	0	0
other Ada concepts	0	0	0	0

TABLE 3-9h
SYSTEM INTEGRATION MANAGER/RESEARCH STAFF
SELF-DESCRIBED TITLES

Programmer
Computer Specialist
Computer Specialist
Engineering Manager
Associate Applications Analyst
Manager, Software Development
Programmer Analyst
Computer Specialist
Computer Specialist
*
Software System Engineer
Software Design Specialist
Senior System Engineer

Key: * no title given on survey.

3.4.10 SYSTEM INTEGRATION SENIOR TECHNICAL STAFF

3.4.10.1 Job Summary

The System Integration Senior Technical Staff is an individual who has significant technical responsibility typically in the area of systems analysis. This person has only moderate management responsibility.

3.4.10.2 Summary of Survey Results

The System Engineering Senior Technical Staff category contains people with a variety of experience levels. Forty-one percent have over ten years experience, twenty-three have two to five years, eighteen percent have five to ten years, and sixteen percent have less than two years. The principal outputs and duties of this cluster include the preparation of technical reports and temporary engineering reports, system analysis, conducting design reviews, program management, quality assurance, and configuration management. The most widely used programming languages for this cluster are FORTRAN, Assembler, and BASIC; the most widely used methodologies are structured programming, top-down design, and structured design. The programming constructs and concepts used most frequently by the group are global and local variables, conditional statements, and iteration. This group has a marginal knowledge of Ada programming concepts.

TABLE 3-10a
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	7
TWO TO FIVE YEARS	10
FIVE TO TEN YEARS	8
OVER TEN YEARS	18
TOTAL	43

TABLE 3-10b
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
other support	1.0	debugging or modifying	2.0
prepare technical reports	1.0	documenting code	2.0
prepare temporary engineering reports	1.0	preparing test plans	2.0
updating training manuals	1.0	software module testing	2.0
system analysis	1.0	documenting test results	2.0
technical advice to Configuration Control Board	1.0	preparing trouble reports	2.0
conduct support design reviews	1.0	define data structures and algorithms	2.0
other development	1.0	design	2.0
quality assurance	1.0	code	2.0
configuration management	1.0	library control	2.0
other administrative tasks	1.0	preparing field engineering reports	2.0
teaching	1.0	preparing schedules	2.0
program management	1.5	technical management	2.0
preparing budgets	1.5	monitoring contracts	2.0
formulation strategy	1.5	interviewing personnel	2.0
functional system design	1.5	support analysis	2.0
formulating policy	1.5	support design	2.0
support quality assurance	1.5	system software testing	2.0
sales marketing	1.5	code/patch	2.0
being trained	1.5	defining test cases	2.0
preparing management information reports	1.5	support formulation of policy	2.0
support monitoring contracts	1.5	support program management	2.0
maintaining configuration procedures	1.5	coding	2.0
Software Configuration Control Board participation	1.5	support configuration management	2.0
attend support walkthroughs	1.5	contract negotiation	2.0
attend support design reviews	1.5	functional module design	2.0
defined subsystem interfaces	2.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-10c
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	6
CMS-2	14
C	2
FORTRAN	40
COBOL	20
ASSEMBLER	38
PL/I	17
PASCAL	16
BASIC	27
ALGOL	4
RATFOR,WATFOR,WATFIV	8
MODULA	1
SIMULA	0
XPL	0
MMP	0
FORTH	4
Ada	14
LISP	4
SNOBOL	6
ECL	0
GPSS	7
SAS	2
PROTEGE	0
PPL	0
APL	12
Other	7

TABLE 3-10d
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	6	11	2	0
SADT	2	2	4	0
SREM	3	3	1	0
HIPO	7	9	10	2
Jackson Design	3	2	0	0
Structured Design	1	8	14	13
Warnier/Orr Design	4	3	1	0
N-S/Chapin Charts	2	2	0	0
Beamson Tables	1	1	0	0
Program Design Language	7	11	12	5
Structured Programming	0	9	16	18
Structured Walkthroughs	2	14	10	11
Top-Down Design	0	6	20	16
Top-Down Testing	1	17	10	9
Bottom-Up Design	4	18	5	4
Bachman Diagramming	1	3	0	0
Entity Diagrams	0	2	1	0
Data Abstraction	2	7	6	0
Other methodology	0	0	0	0

TABLE 3-10e
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Constructs \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	5	4	9	5
floating point types	1	9	21	11
fixed point types	1	6	19	14
user defined types	5	7	12	7
pointers	1	7	17	13
typed pointers	7	11	6	6
ranges	5	8	14	7
records	2	8	15	15
variant records	3	9	8	6
object/type declarations	5	10	13	10
global variables	1	3	20	18
local variables	1	3	19	20
formal and actual parameters	4	9	9	12
reserved words	0	7	14	16
blocks	0	9	18	9
case statement	2	4	17	15
if/then/else statements	0	2	16	24
loop/for/while/until statements	0	5	16	20
exit statements (for loops)	0	5	19	15
procedures	1	6	17	17
functions	1	4	18	17
return statements	0	2	17	24
clusters/modules/packages	6	7	17	6
stubs	2	7	13	8
goto statements	0	7	19	16
comments	0	1	14	27
exception handlers	4	10	11	8
tasks/coroutines	3	11	8	4
other programming constructs	0	1	0	0

TABLE 3-10f
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	6	4	2	1
data encapsulation (compools)	2	10	5	2
name scoping	6	8	3	0
name visibility	4	7	3	0
static and dynamic nesting	9	12	8	1
iteration	3	5	17	12
conditional statements	0	4	14	23
recursion	3	11	13	9
concurrency	7	12	10	2
strong typing	3	8	6	5
type conversion	3	7	11	5
data abstraction	5	9	6	1
generics	4	9	5	0
loop invariants	4	9	4	1
parameter binding	6	10	2	1
version numbers	2	10	7	5
other programming concepts	0	0	0	0

TABLE 3-10g
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
ADA PROGRAMMING CONCEPTS

Ada Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	6	6	4	0
user-defined types	4	11	6	0
subtypes	5	17	2	0
derived types	5	4	3	0
real types	4	14	3	3
floating point types	3	16	3	3
fixed point types	3	14	4	3
record types	3	12	3	3
record types with discriminants	5	4	2	1
slices	3	6	1	0
aggregates	3	8	1	0
allocators	2	8	1	0
access types	4	7	1	0
overloading	4	9	1	0
packages	6	6	3	1
private types	5	8	1	0
scope	3	7	2	0
short circuiting	3	4	0	1
visibility	5	5	3	0
tasking	7	10	3	2
task types	7	9	1	0
rendezvous	3	8	2	0
entries	3	8	3	0
entry families	2	5	1	0
separate compilation	1	13	5	2
exceptions	4	12	3	0
generic program units	5	8	1	0
instantiation	5	6	1	0
elaboration	1	4	2	0
context specification	2	8	1	0
information hiding	3	10	1	1
mutual recursion	6	5	1	0
other Ada concepts	0	0	0	0

TABLE 3-10h
SYSTEM INTEGRATION SENIOR TECHNICAL STAFF
SELF-DESCRIBED TITLES

Programmer
Analyst
*
Electrical Engineer
Principle Programmer Analyst
System Engineer
Senior Engineering Specialist
Software Engineer
*
*
Software System Programmer
*
*
*
Programming Aide
Associate Scientific Programmer
*
*
*
Senior System Analyst
*
Software Design Specialist
Software Engineering Specialist
Software Engineer
Software Engineer
Principle Engineer
*
Software Engineer
*
General Engineer
Software Quality Assurance
Software Engineer
Computer Specialist
Software Engineer
Senior Dynamics Engineer
Computer Specialist
Software Design Specialist
*
Engineer
Computer Systems Analyst
Software System Analyst
Computer Specialist
Mathematician

Key: * no title given on survey.

3.4.11 SYSTEM INTEGRATION ENGINEER

3.4.11.1 Job Summary

The System Integration Engineer has no management responsibility. This person's principal activities involve design, coding, testing, etc.

3.4.11.2 Summary of Survey Results

The System Integration Engineer is typically an individual with two to five years experience, whose principal outputs and duties include technical walkthroughs, coding and patching, defining system interfaces, defining text cases, software module testing, and conducting and attending design reviews. This individual is most familiar with FORTRAN and Assembler, and has frequently used structured programming, top-down design, and structured design. The programming concepts and constructs most familiar to the System Integration Engineer are global and local variables, conditional statements, iteration, procedures, and functions. The group has a marginal knowledge of Ada programming concepts.

TABLE 3-11a
SYSTEM INTEGRATION ENGINEER
YEARS OF SOFTWARE DEVELOPMENT OR SUPPORT

YEARS OF INVOLVEMENT	RESPONDENTS
LESS THAN TWO YEARS	13
TWO TO FIVE YEARS	32
FIVE TO TEN YEARS	14
OVER TEN YEARS	14
TOTAL	73

TABLE 3-11b
SYSTEM INTEGRATION ENGINEER
PRINCIPAL OUTPUTS AND DUTIES

Outputs and Duties	Cluster Rank	Outputs and Duties	Cluster Rank
technical walkthroughs	1.0	documenting test results	1.0
code/patch	1.0	prepare trouble reports	1.0
conduct support walkthroughs	1.0	analyzing trouble reports	1.0
support design	1.0	attend walkthroughs	1.0
functional module design	1.0	design	1.0
define global data structures	1.0	conduct design reviews	1.0
define subsystem interfaces	1.0	attend design reviews	1.0
define data structures and algorithms for own use	1.0	code	1.0
coding	1.0	attend support design reviews	1.5
debugging or modifying code	1.0	attend support walkthroughs	1.5
prepare system requirement documents	1.0	formulating strategy	1.5
support analysis	1.0	prepare redlined documentation	1.5
prepare version description manuals	1.0	functional system design	1.5
prepare user manuals	1.0	being trained	1.5
documenting code	1.0	reviewing technical work	2.0
defining test cases	1.0	teaching	2.0
prepare test drivers	1.0	support technical management	2.0
prepare test plans	1.0	attend requirements reviews	2.0
system software test	1.0	conduct support design reviews	2.0
define module test cases	1.0	support configuration management	2.0
software module testing	1.0		

Key:

1.0 indicates that this cluster as a group was involved more with this activity than any other cluster.

1.5 indicates that this cluster as a group was tied with another cluster for the 1.0 rank.

2.0 indicates that this cluster as a group was involved with this activity second to another cluster.

TABLE 3-11c
SYSTEM INTEGRATION ENGINEER
PROGRAMMING LANGUAGES
(STUDIED OR USED)

LANGUAGE	RESPONSES
JOVIAL	12
CMS-2	21
C	4
FORTRAN	68
COBOL	31
ASSEMBLER	63
PL/I	44
PASCAL	41
BASIC	44
ALGOL	17
RATFOR,WATFOR,WATFIV	21
MODULA	1
SIMULA	2
XPL	2
MMP	0
FORTH	6
Ada	11
LISP	12
SNOBOL	15
ECL	1
GPSS	9
SAS	2
PROTEGE	1
PPL	0
APL	25
Other	13

TABLE 3-11d
SYSTEM INTEGRATION ENGINEER
KNOWLEDGE OF METHODOLOGIES

Methodology \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
PSL/PLA	7	14	1	0
SADT	6	3	0	0
SREM	5	1	0	0
HIPO	6	19	12	3
Jackson Design	4	3	2	1
Structured Design	2	10	19	36
Warnier/Orr Design	6	6	3	0
N-S/Chapin Charts	3	7	3	1
Beamson Tables	0	0	0	0
Program Design Language	7	15	12	29
Structured Programming	2	5	10	56
Structured Walkthroughs	8	15	26	11
Top-Down Design	1	9	18	43
Top-Down Testing	3	20	24	17
Bottom-Up Design	3	32	12	11
Bachman Diagramming	1	2	0	1
Entity Diagrams	0	2	0	1
Data Abstraction	6	8	5	4
Other methodology	0	0	0	1

TABLE 3-11e
SYSTEM INTEGRATION ENGINEER
KNOWLEDGE OF PROGRAMMING CONSTRUCTS

Constructs \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	5	11	9	11
floating point types	1	10	19	40
fixed point types	0	5	18	45
user defined types	2	16	13	30
pointers	1	9	16	46
typed pointers	5	16	10	16
ranges	4	17	15	26
records	2	13	19	35
variant records	5	17	11	14
object/type declarations	2	13	18	26
global variables	0	3	11	59
local variables	0	2	10	61
formal and actual parameters	3	7	14	38
reserved words	1	11	10	44
blocks	2	11	12	42
case statement	2	9	11	48
if/then/else statements	0	6	9	58
loop/for/while/until statements	0	6	10	57
exit statements (for loops)	0	16	17	39
procedures	1	2	14	53
functions	0	4	15	52
return statements	0	3	12	57
clusters/modules/packages	4	10	18	25
stubs	1	18	13	20
goto statements	0	18	23	32
comments	0	0	7	66
exception handlers	3	18	12	21
tasks/coroutines	2	24	11	15
other programming constructs	0	0	0	1

TABLE 3-11f
SYSTEM INTEGRATION ENGINEER
KNOWLEDGE OF PROGRAMMING CONCEPTS

Concepts \ Knowledge Level	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
importing/exporting names	3	12	2	6
data encapsulation (compools)	9	16	8	7
name scoping	1	10	8	8
name visibility	1	9	6	6
static and dynamic nesting	2	17	7	11
iteration	1	5	19	39
conditional statements	1	1	12	53
recursion	3	16	25	20
concurrency	5	23	10	9
strong typing	4	11	10	9
type conversion	3	14	16	19
data abstraction	9	18	3	9
generics	9	14	4	3
loop invariants	9	14	6	5
parameter binding	7	16	3	7
version numbers	1	14	8	25
other programming concepts	0	0	0	1

TABLE 3-11g
SYSTEM INTEGRATION ENGINEER
ADA PROGRAMMING CONCEPTS

Knowledge Level / Ada Concepts	Have Heard of	Know What Concept Is	Have Used Moderately	Have Used Frequently
enumeration types	3	12	3	3
user-defined types	4	16	6	6
subtypes	5	12	2	4
derived types	6	10	2	1
real types	1	20	4	11
floating point types	1	21	6	9
fixed point types	1	20	5	9
record types	3	19	5	9
record types with discriminants	4	7	5	2
slices	3	2	2	1
aggregates	5	7	3	1
allocators	2	9	2	1
access types	2	7	4	2
overloading	0	10	3	0
packages	3	10	3	3
private types	3	11	3	1
scope	2	15	2	5
short circuiting	3	4	1	1
visibility	2	11	1	4
tasking	4	19	1	3
task types	6	13	2	0
rendezvous	4	6	1	1
entries	3	12	2	2
entry families	6	4	1	1
separate compilation	1	20	4	4
exceptions	1	14	5	3
generic program units	3	10	2	1
instantiation	3	6	2	1
elaboration	3	2	1	2
context specification	5	1	1	0
information hiding	2	13	2	3
mutual recursion	3	4	0	1
other Ada concepts		0	0	0

TABLE 3-11h
SYSTEM INTEGRATION ENGINEER
SELF-DESCRIBED TITLES

Senior Software Engineer	Software Engineer
Software Engineer	Senior Software engineer
Software Engineer	Software Engineer
Software Engineer	*
Software Design Specialist	Software Engineer
Software Design Specialist	Software Engineer
*	*
*	*
Software Engineer	Software Engineer
Consultant	*
Computer Scientist	Advanced Research Engineer
Software Engineering Specialist	System Programming Analyst
Software Design Specialist	*
*	Engineer
*	Senior Software Engineer
*	Software Engineer
*	Engineer
Associate Engineer	Engineer
Software Engineer	Software Engineer
*	*
Electrical Engineer	*
*	*Programmer*
*	Programmer Analyst
Scientific Programmer	Programmer
*	Principle Programmer
Programmer	Programmer Analyst
Software Engineer	*
*	*
Senior Engineer	Senior Engineer
Software Engineer	*
R&D Engineer	*
Engineering Specialist	Software Engineer
Senior Engineer	*
Senior Engineer	Software Engineer
Programmer	R&D Engineer
Software Engineer	Senior Software Engineer
	Senior Engineer

Key: * no title given on survey.

Section 4

CURRICULUM TREE

4.1 Introduction

The Curriculum Tree is a graphic representation of the background and "Ada viewpoint" of each generic job catetory identified by analyzing the Industry/Government Work Force Survey (see Section 3). The format of the tree was specified in the Statement of Work for the contract (see Figure 3-3). In establishing the representative background for each job category, responses to the following survey questions were examined for all respondents in a particular category:

Questions	Topic
1	Years of Software Development and/or Support
6, 9, and 11	Principal Outputs
7, 10, and 11	Principal Duties
16	Programming Languages Studied or Used
18A	Knowledge of Methodologies
18B	Knowledge of Programming Constructs
18C	Knowledge of Programming Concepts
18D	Knowledge of Ada Programming Concepts
-	Self-Described Title

As discussed in Section 3, SofTech's project team then formulated job titles and descriptions consistent with the clusters which emerged from the data analysis. Finally, with the backgrounds, job titles and descriptions in place, the project team made determinations regarding an "Ada Viewpoint" for each category. The Ada Viewpoint represents what a person needs to know about the Ada Programming Support Environment, the Ada Language, and software engineering methodologies in order to function in a particular job category.

4.2 Job Category Background and Ada Viewpoint

The following figures show the representative background and Ada viewpoint for each generic job category. Owing to the comprehensive nature of the background data it is impractical to reproduce it again here. Therefore, the reader is referred to the tables in Section 3 which contain the background data for each job category.

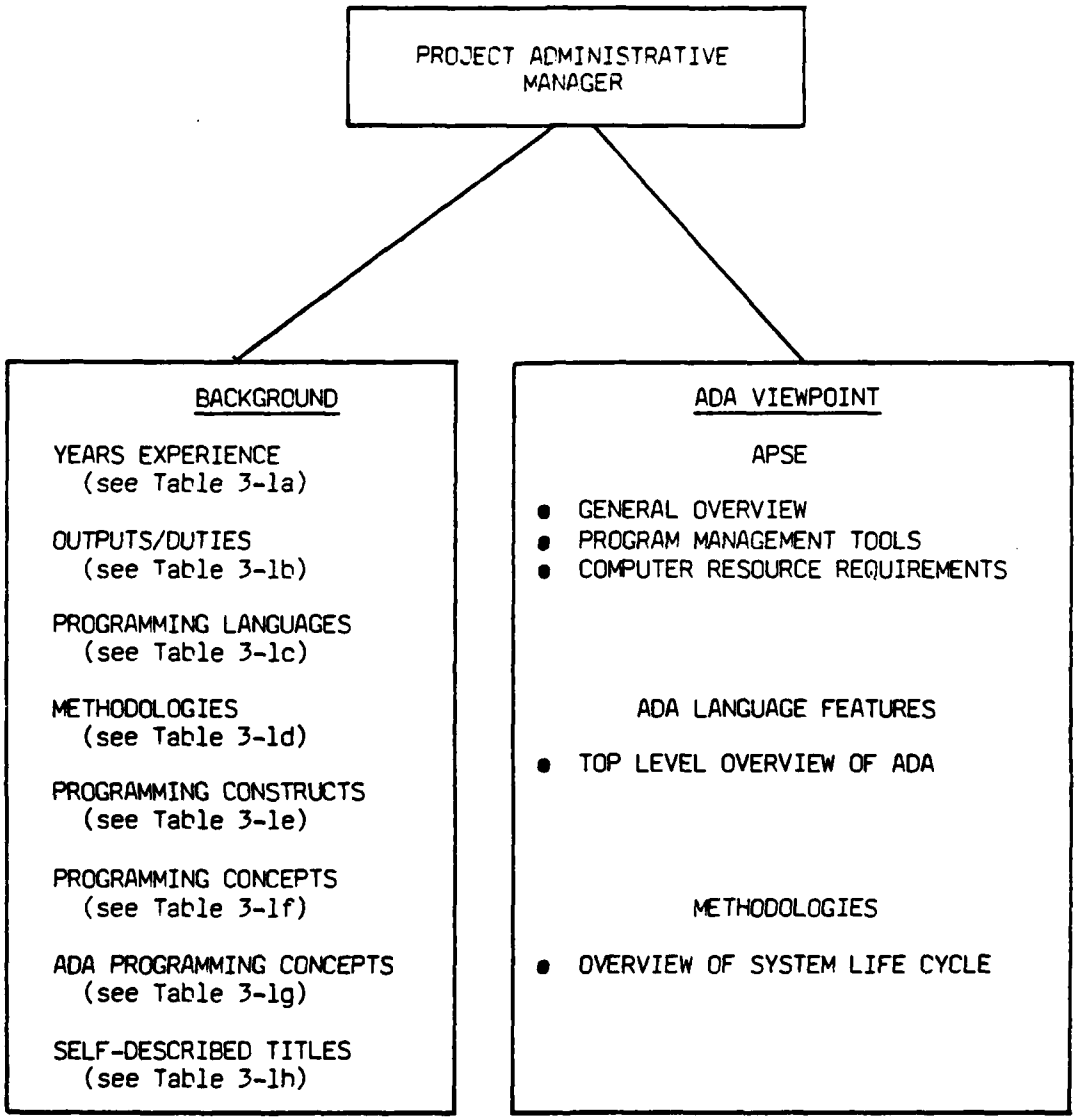


Figure 4-1. Curriculum Tree for Project Administrative Manager

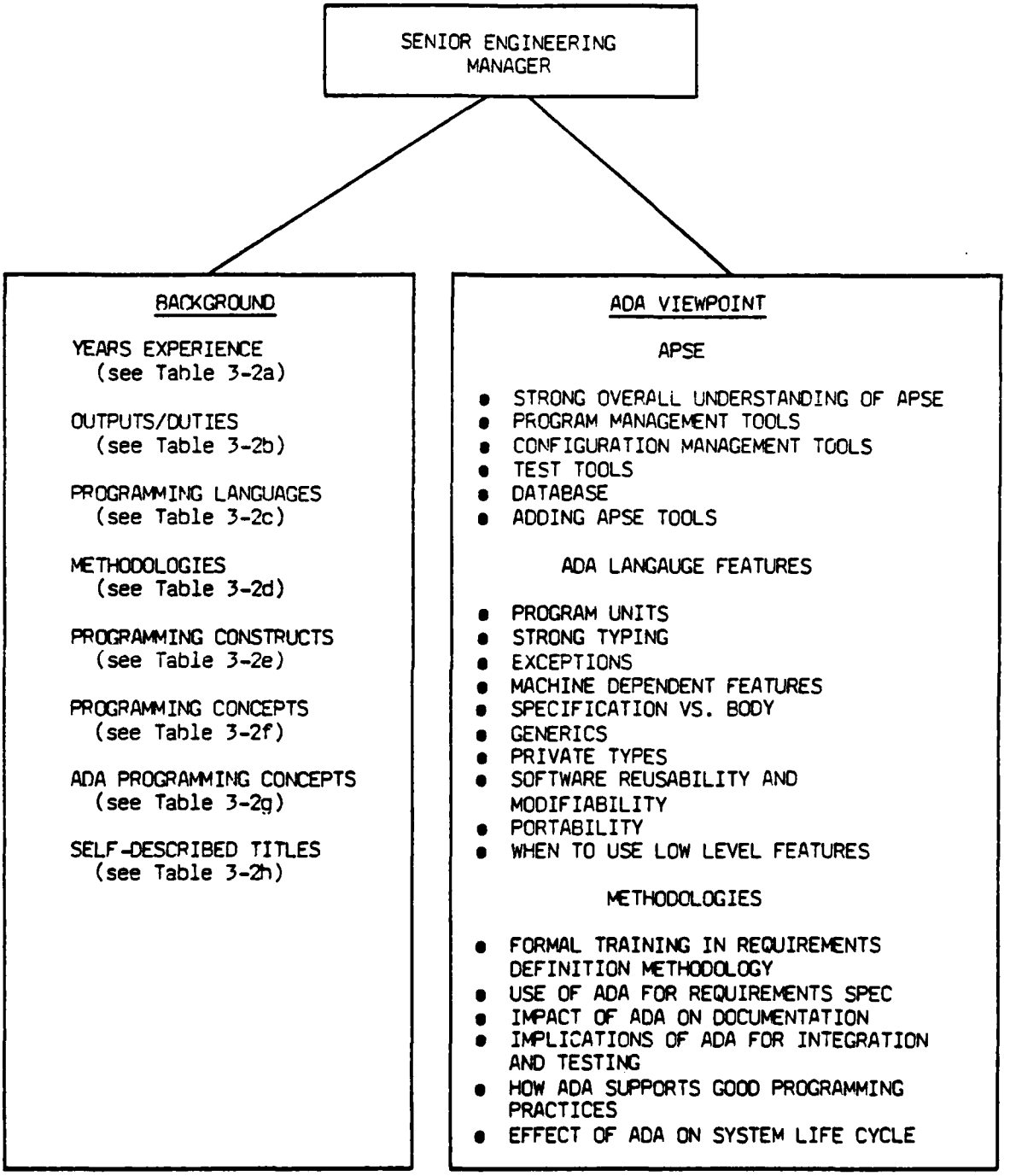


Figure 4-2. Curriculum Tree for Senior Engineering Manager

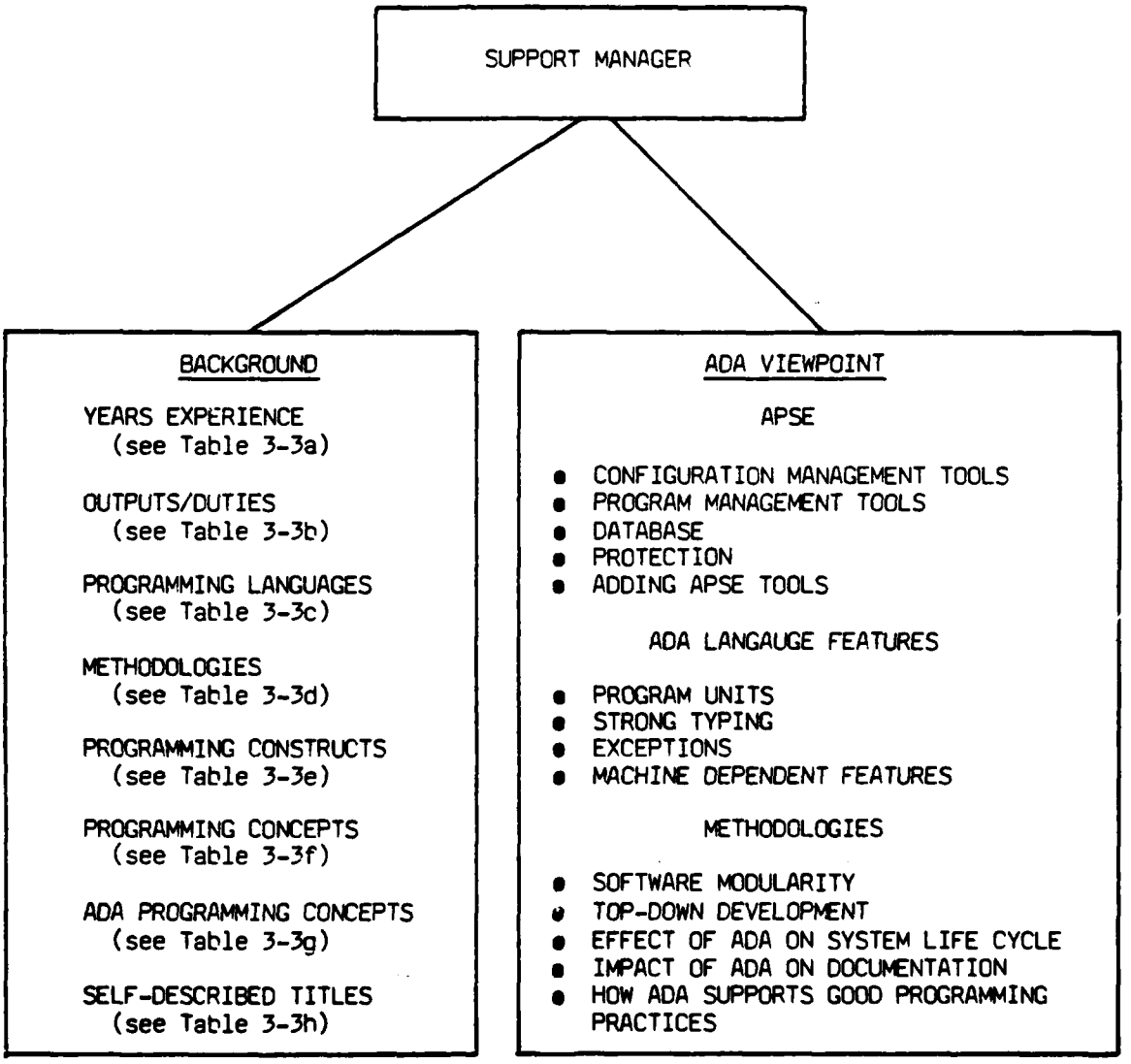


Figure 4-3. Curriculum Tree for Support Manager

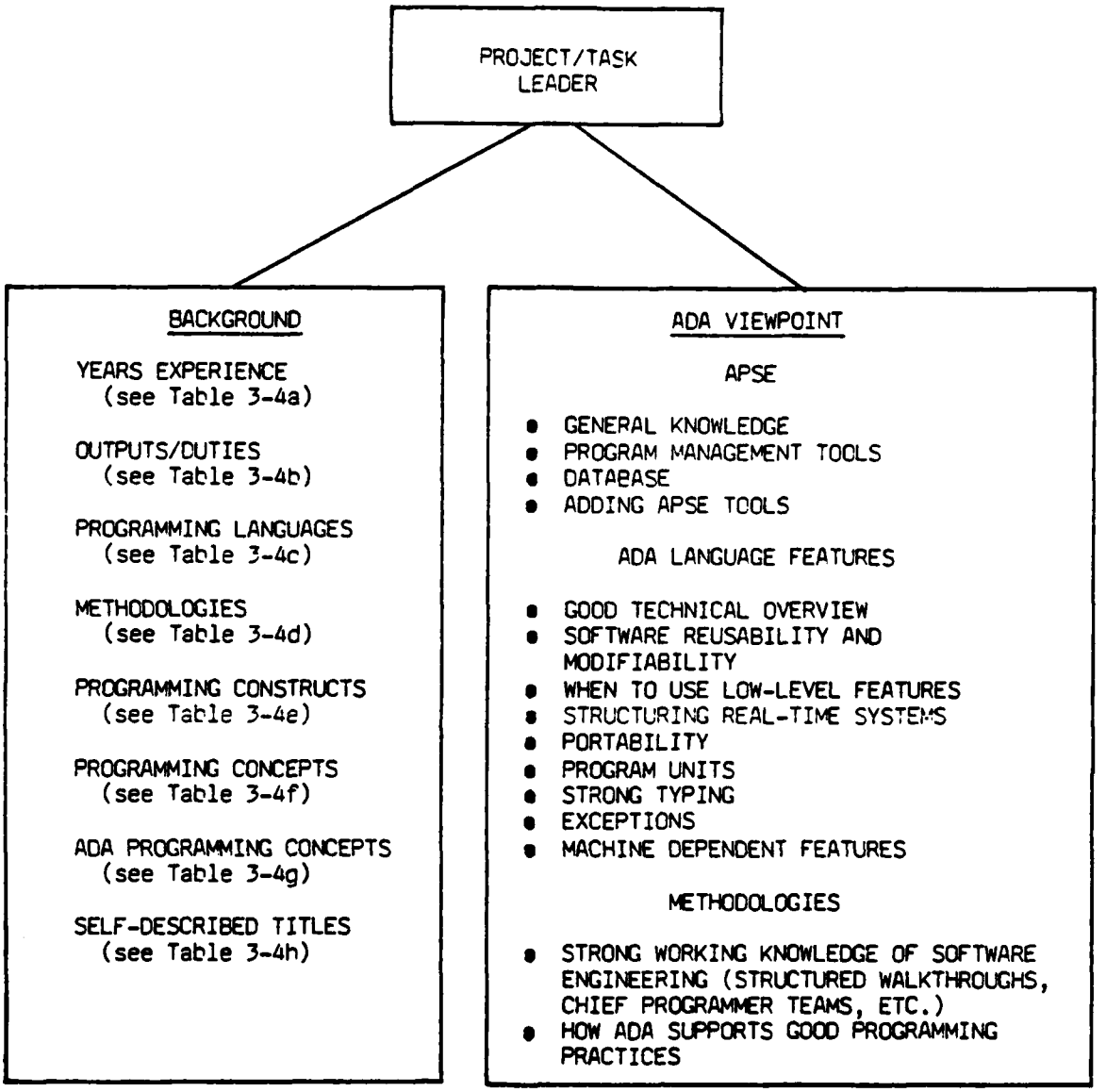


Figure 4-4. Curriculum Tree for Project/Task Leader

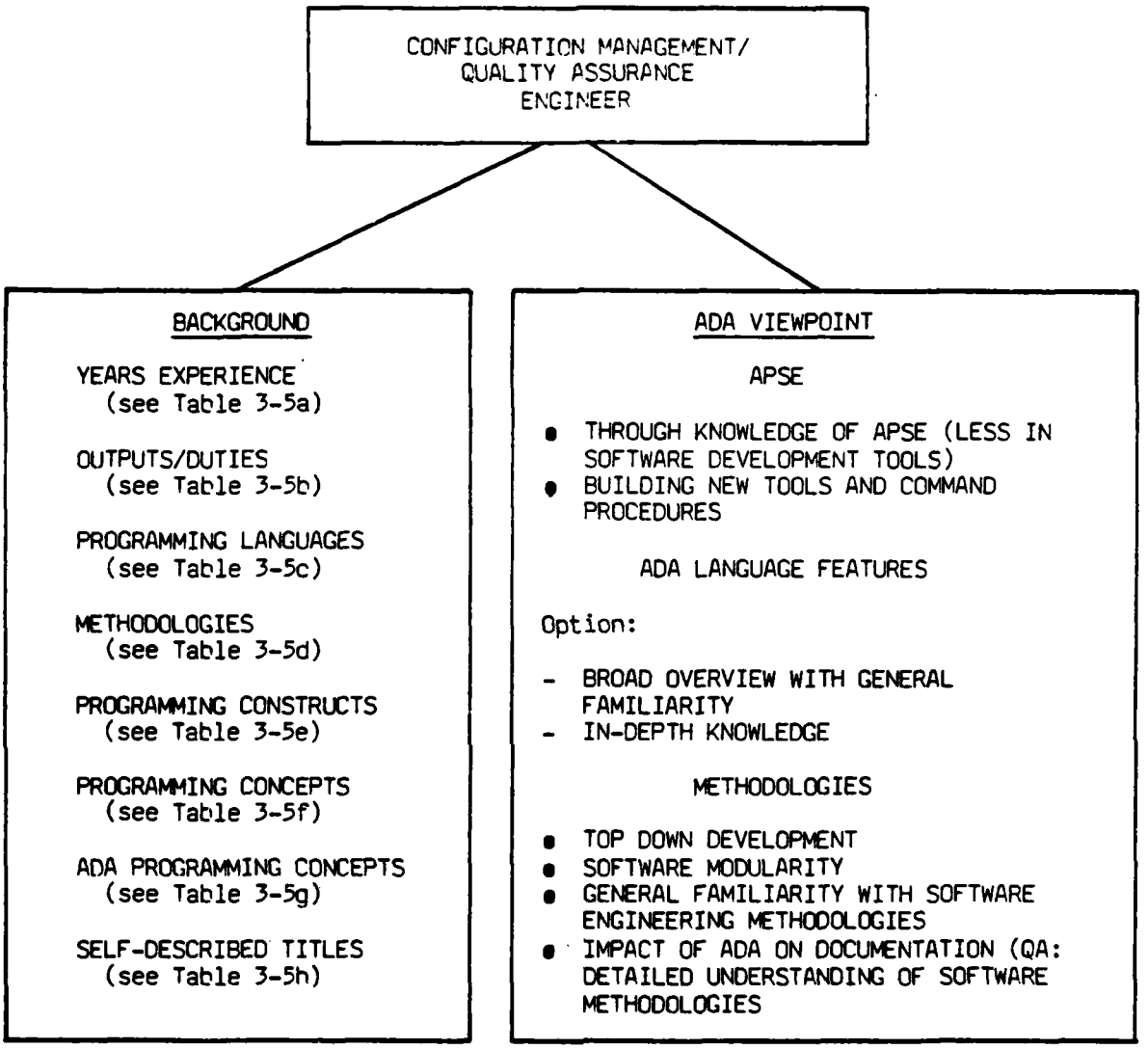


Figure 4-5. Curriculum Tree for Configuration Management/ Quality Assurance Engineer

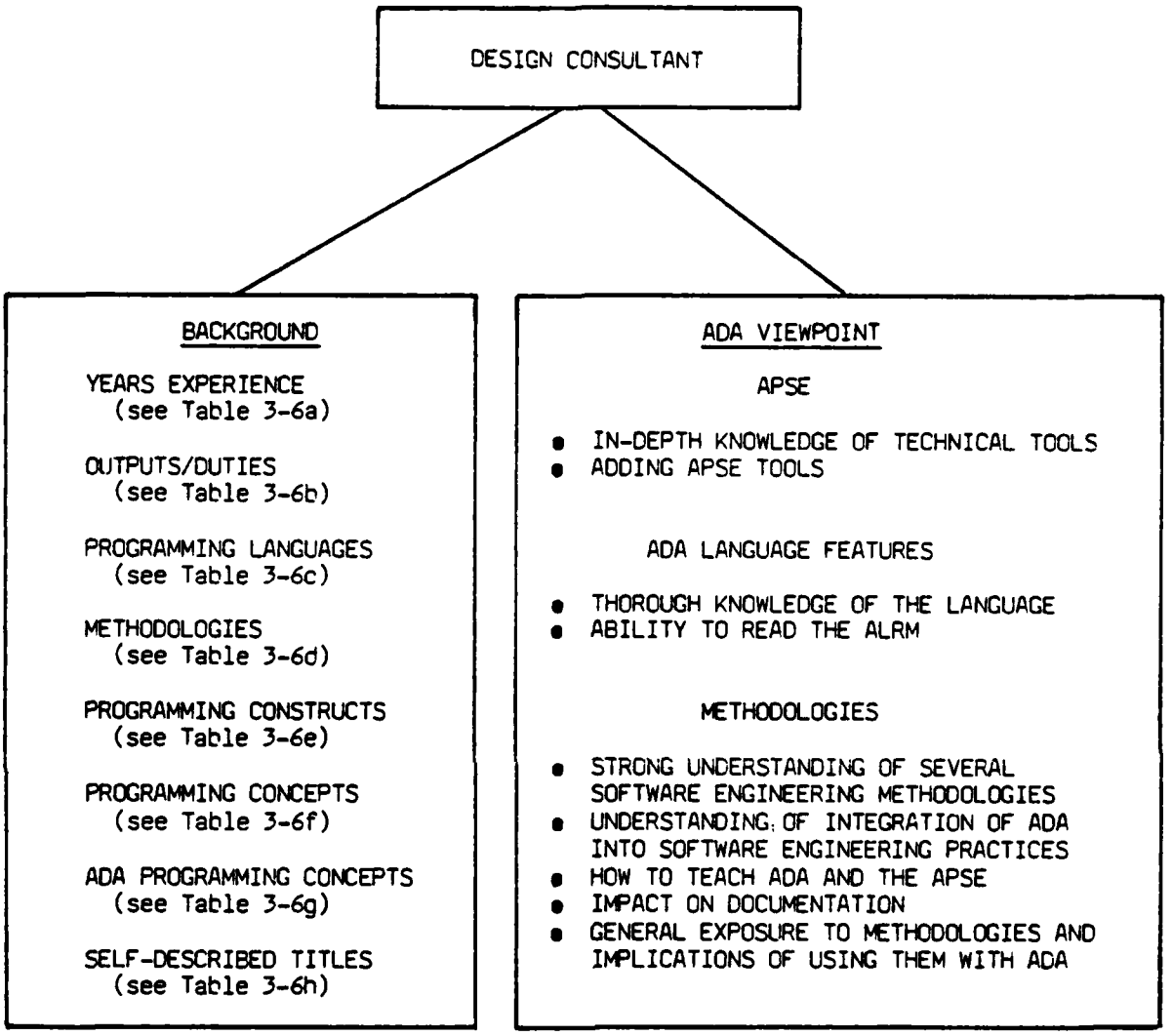


Figure 4-6. Curriculum Tree for Design Consultant

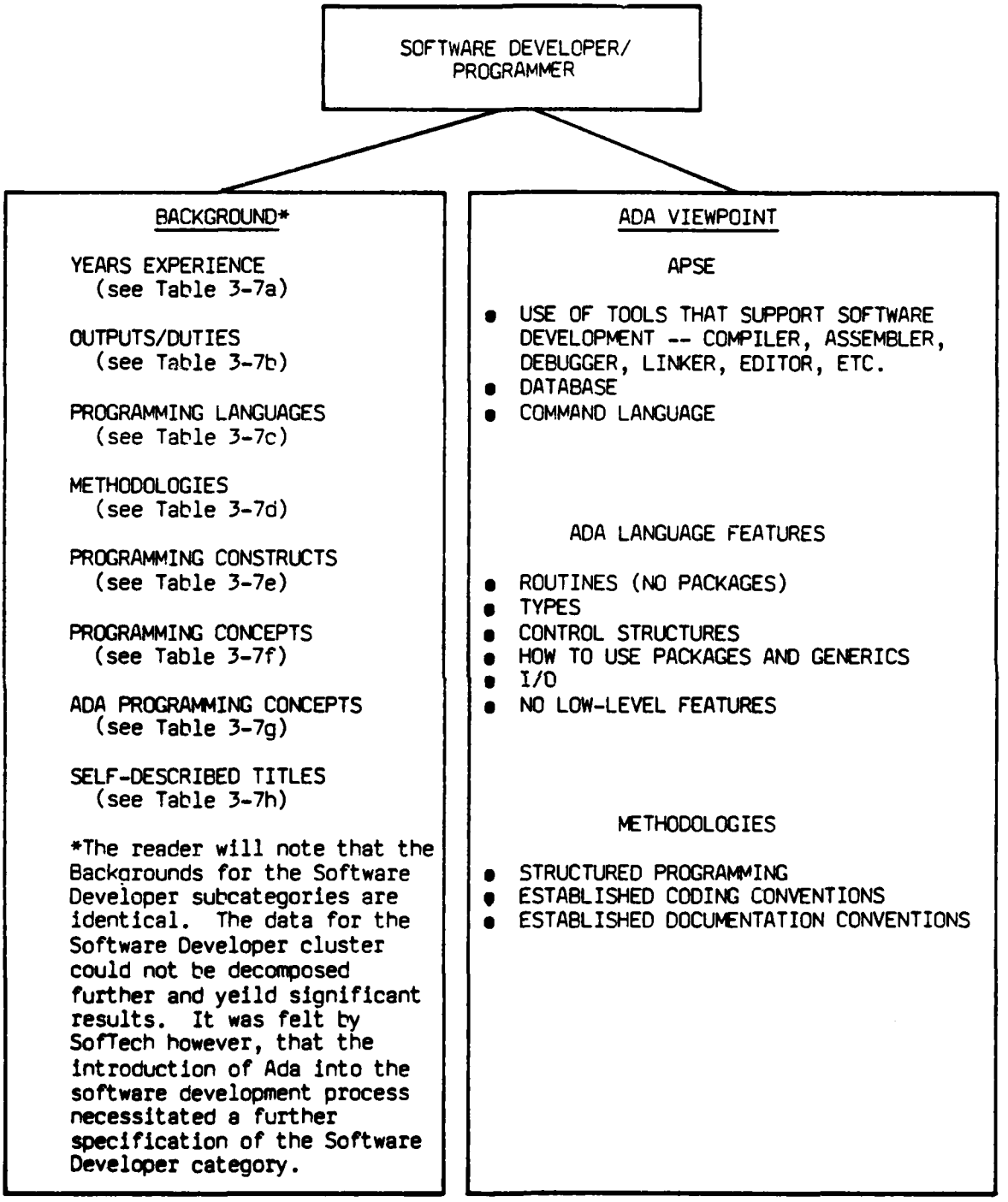


Figure 4-7. Curriculum Tree for Software Developer/Programmer

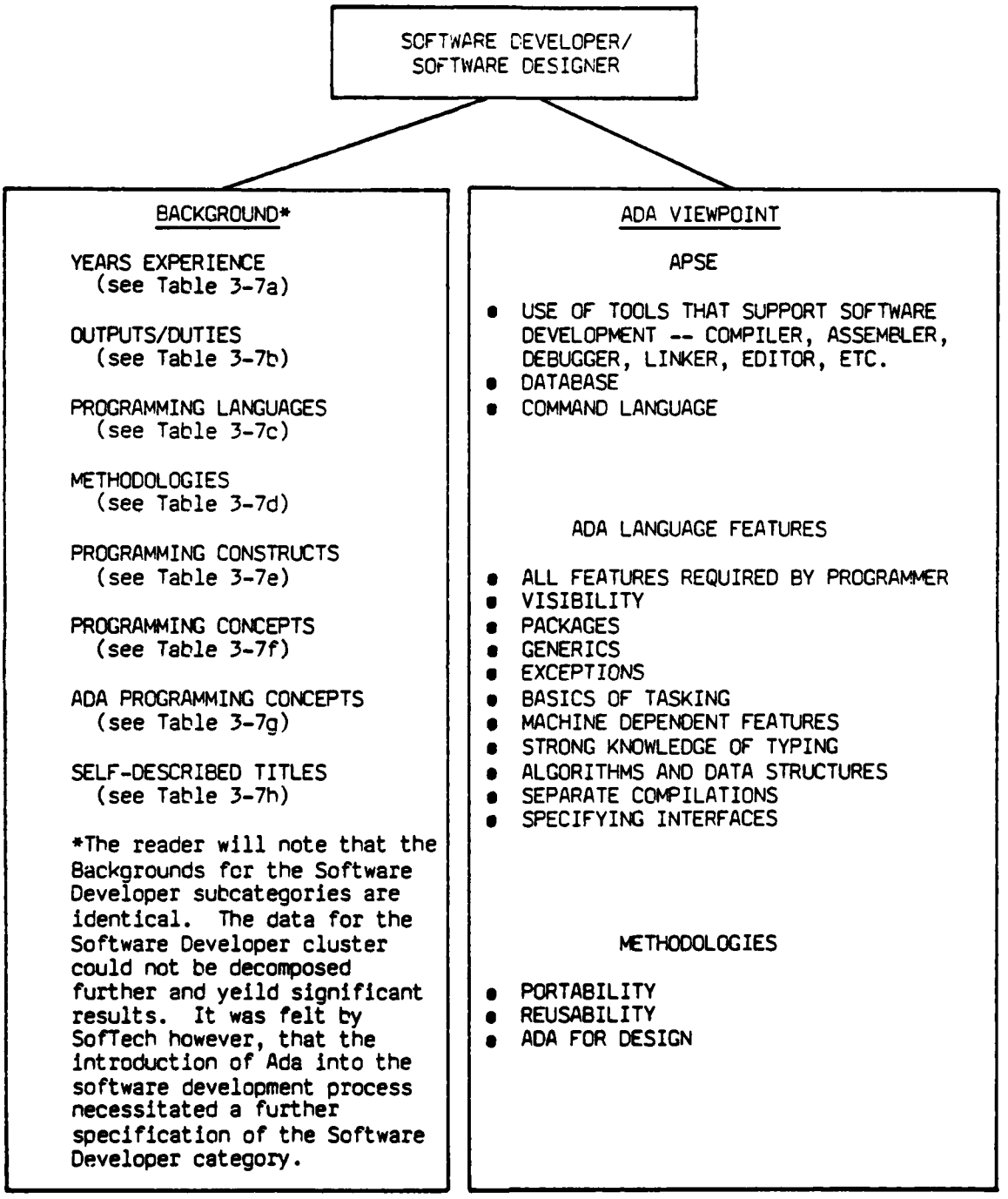


Figure 4-8. Curriculum Tree for Software Developer/Software Designer

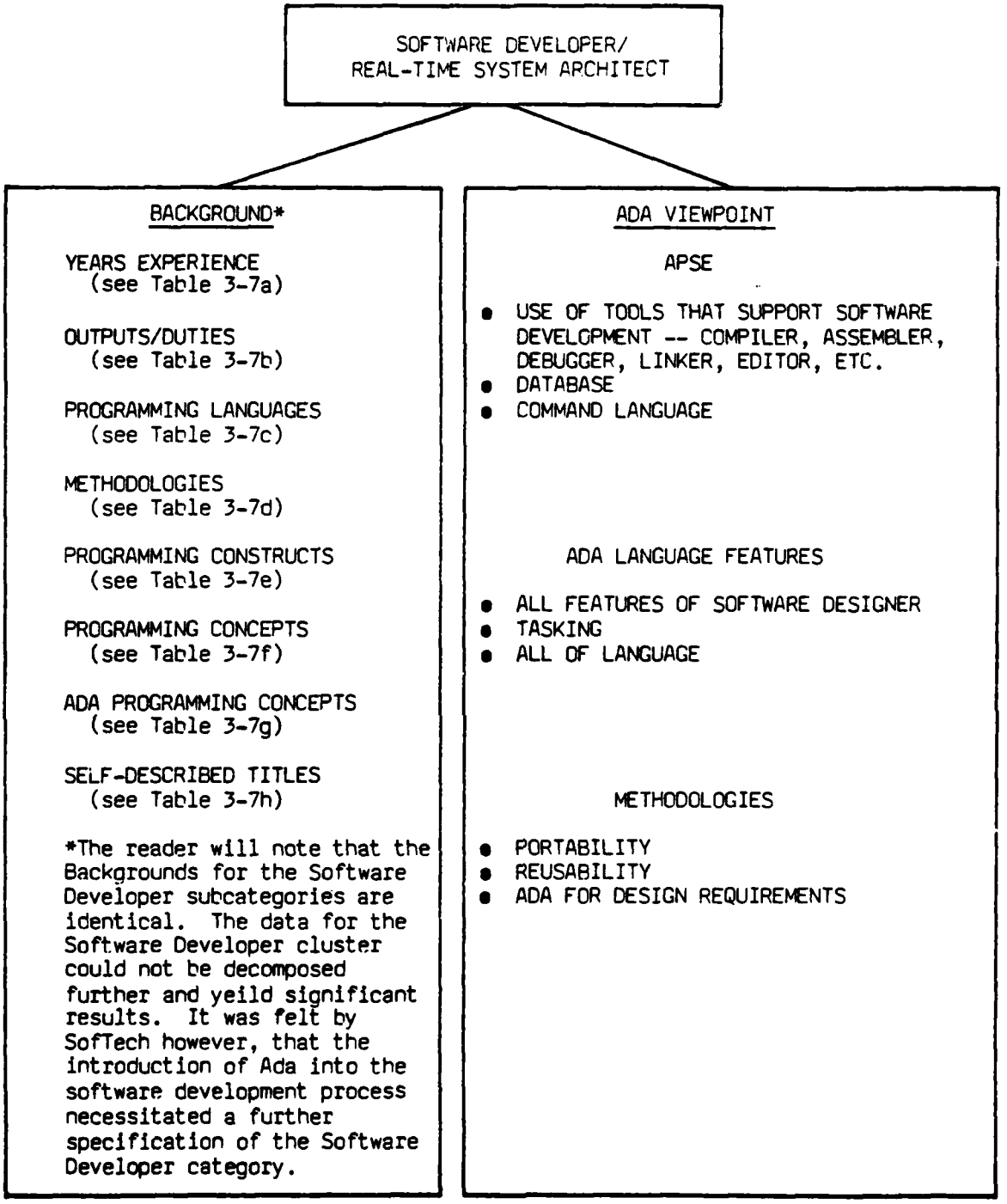


Figure 4-9. Curriculum Tree for Software Developer/ Real-Time System Architect

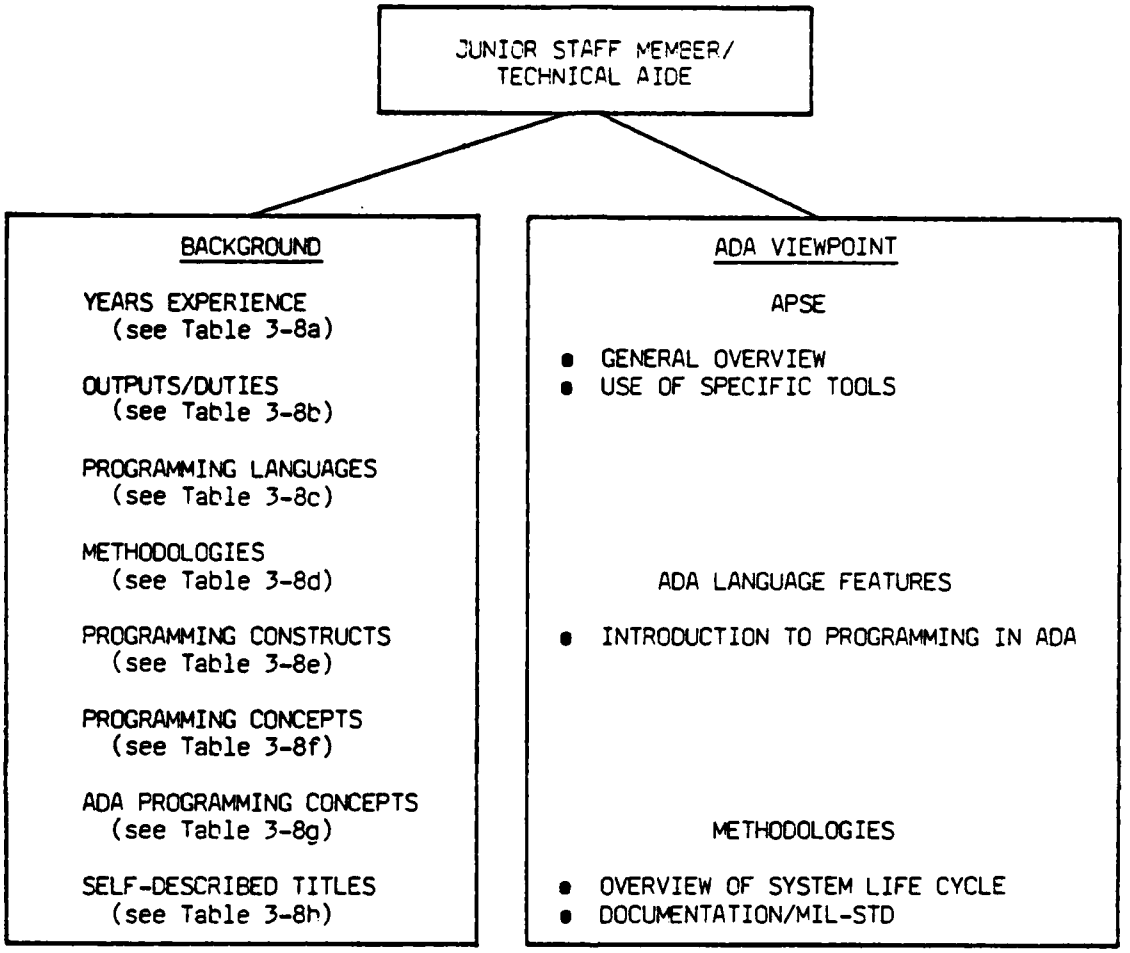


Figure 4-10. Curriculum Tree for Junior Staff Member/Technical Aide

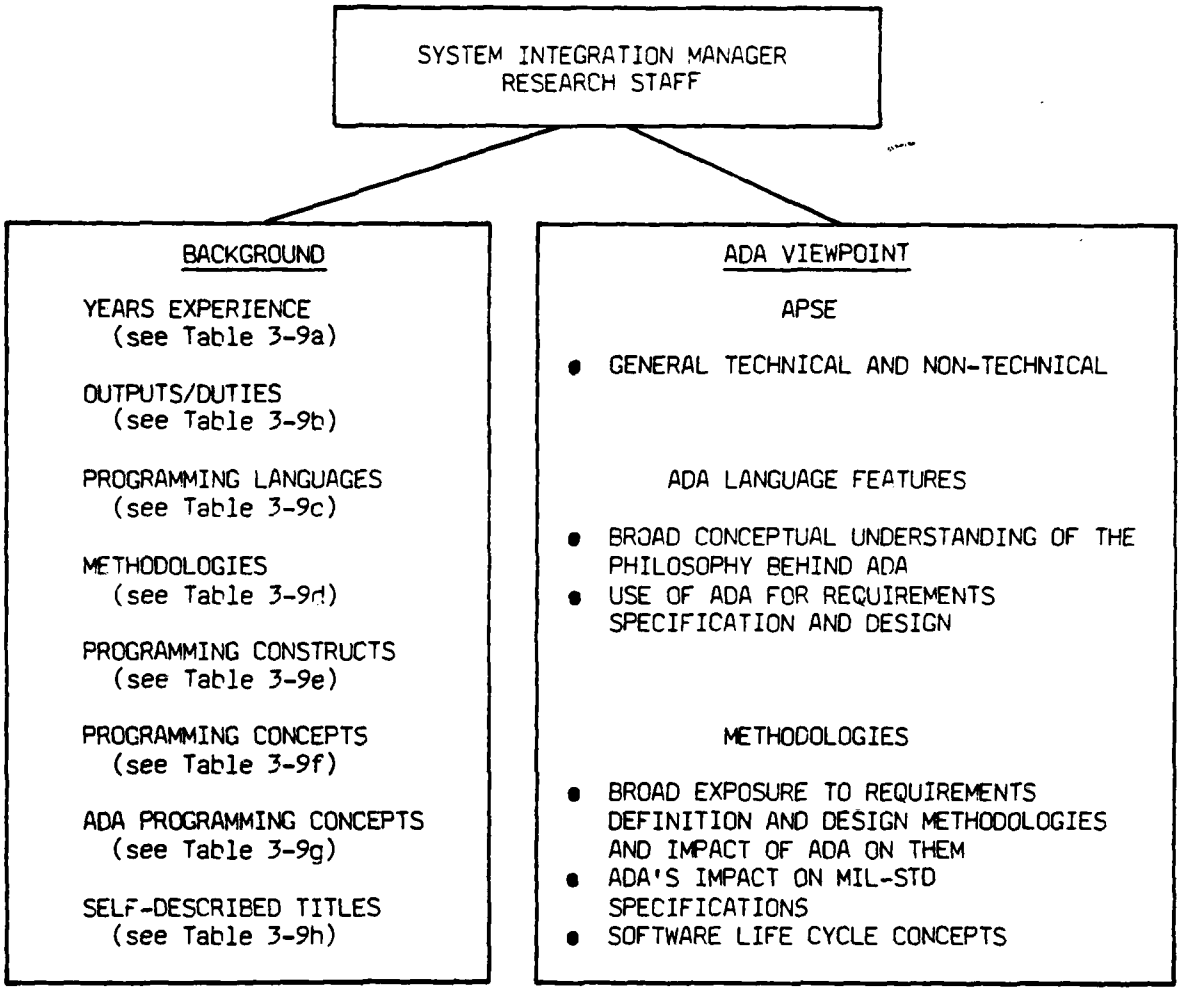


Figure 4-11. Curriculum Tree for System Integration Manager/ Research Staff

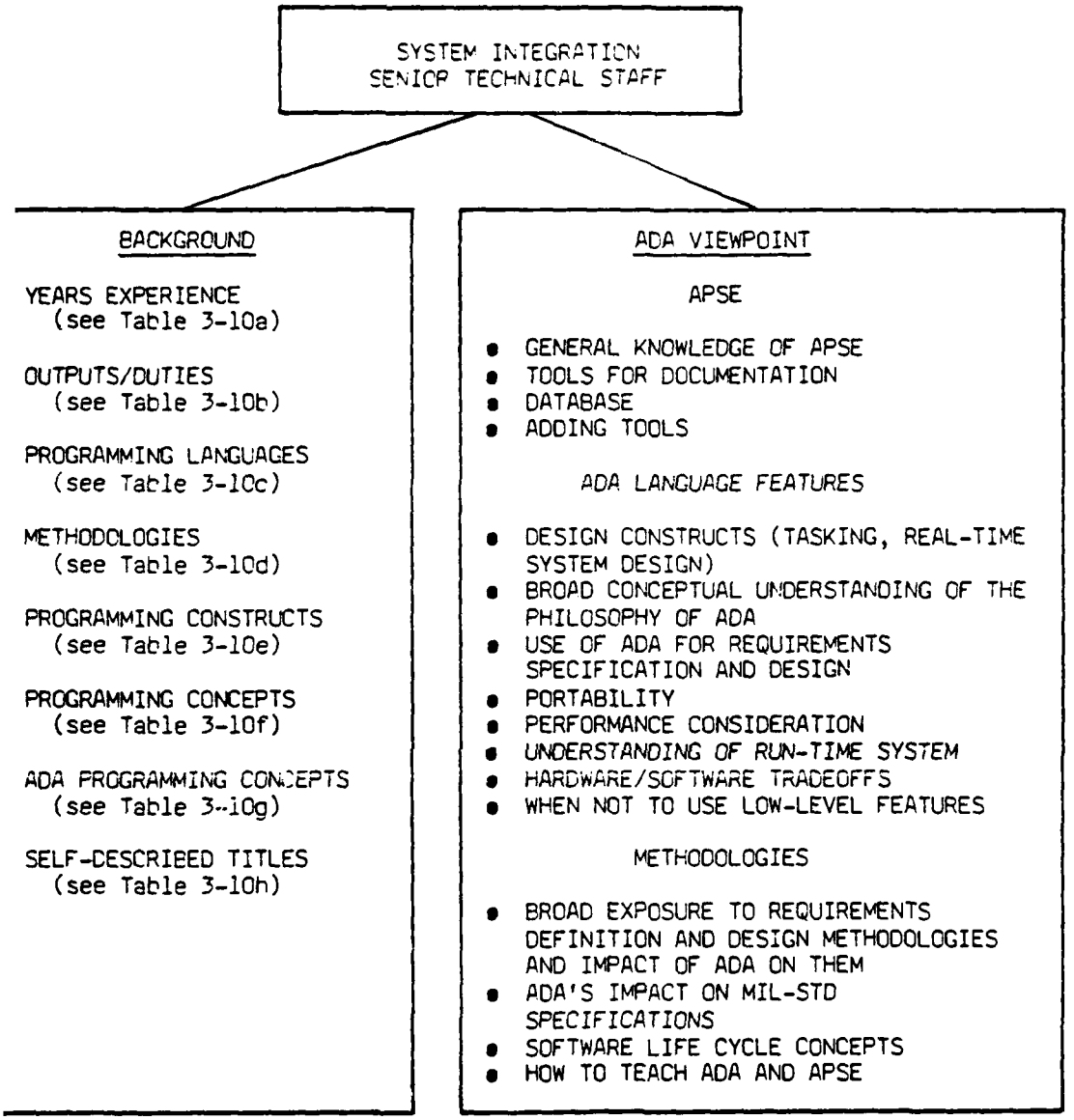


Figure 4-12. Curriculum Tree for System Integration Senior Technical Staff

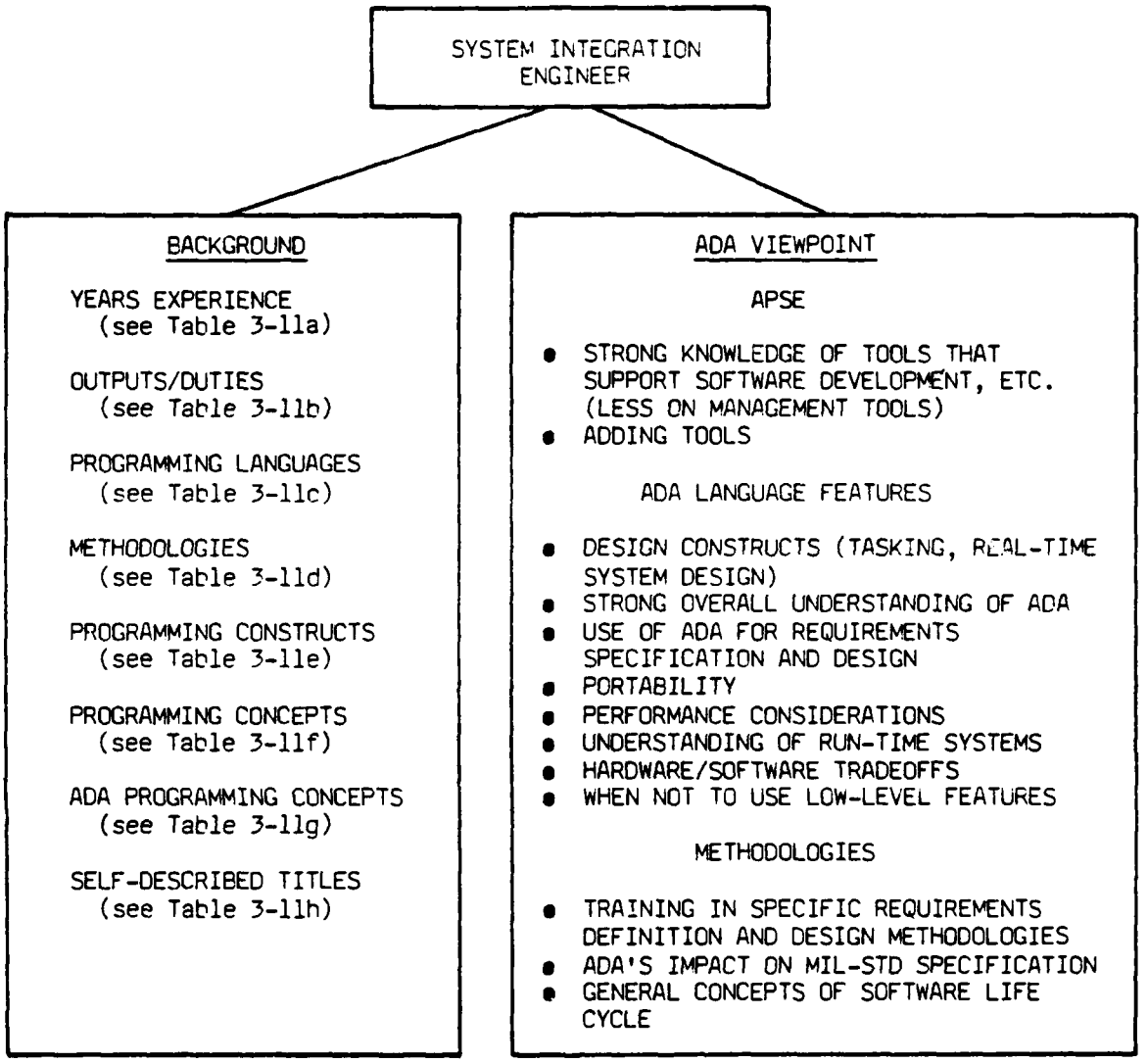


Figure 4-13. Curriculum Tree for System Integration Engineer

Section 5

MODEL ADA TRAINING PROGRAM

5.1 Introduction

SofTech's Curriculum Tree identified a number of job categories, each with different Ada knowledge requirements and thus, different training needs. The Model Ada Training Program outlined below attempts to establish a curriculum for Ada Training which addresses the following issues:

1. For the goals of the Army's Ada Program to be realized, the embedded systems work force must be trained not only in the Ada language but in the Ada Programming Support Environment and software engineering methodologies as well.

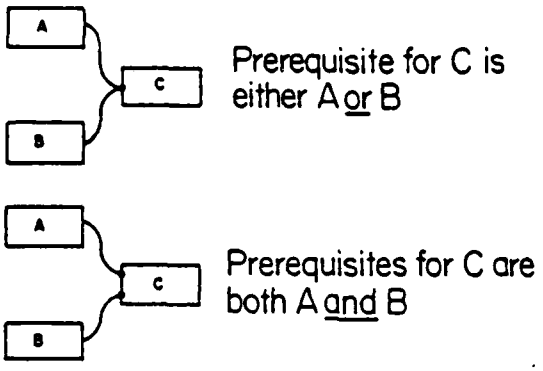
2. Individuals within the same job category will undoubtedly have varying backgrounds and training requirements. While training recommendations for a category are made with the most representative background in mind, the curriculum itself should be structured such that it can be tailored to individual needs.

3. To ensure the acceptance of an Ada Training Program by industry, such a program must be cost-effective. Basically, this translates into a curriculum which does not make unreasonable long term demands on staff time and which allows students to become productive in the shortest possible time.

SofTech has recommended an Ada curriculum comprising thirty-four modular units (plus specialty courses) covering three areas: the Ada Programming Support Environment, the Ada language, and software engineering methodologies. The curriculum identifies prerequisites for appropriate modules and is designed to accommodate a variety of student backgrounds.

Figure 5-1 provides a graphic description of SofTech's recommended Ada Curriculum. This chart indicates progression through the curriculum and interdependencies between course modules. Boxes shown in blue represent the Ada Programming Support Environment (APSE) course modules, those shown in green represent the Ada Language Course modules, and those shown in red represent the methodology course modules. The chart is to be read left to right and is intended to show course prerequisites

(either prior courses or equivalent knowledge). The flow from left to right is along straight or curved lines - right angles are not intended to be paths. Prerequisites are identified as follows:



A detailed description of each of the curriculum modules is found in subsection 5.2. Graphical representations showing suggested paths through the curriculum for each job category are in contained subsection 5.3. *Tables cross-referencing course modules with job categories*, completed case studies and suggested future case studies are in subsection 5.4.

5.2 Module Descriptions

The following pages provide detailed course module descriptions for SofTech's Model Ada Training Program. Each module is identified by name, area (Ada Programming Support Environment, Ada Language, or Methodology), course number, course duration, prerequisites, objective for the course module, a syllabus, and the generic job categories for which the module is recommended.

SofTech Model Ada Training Curriculum

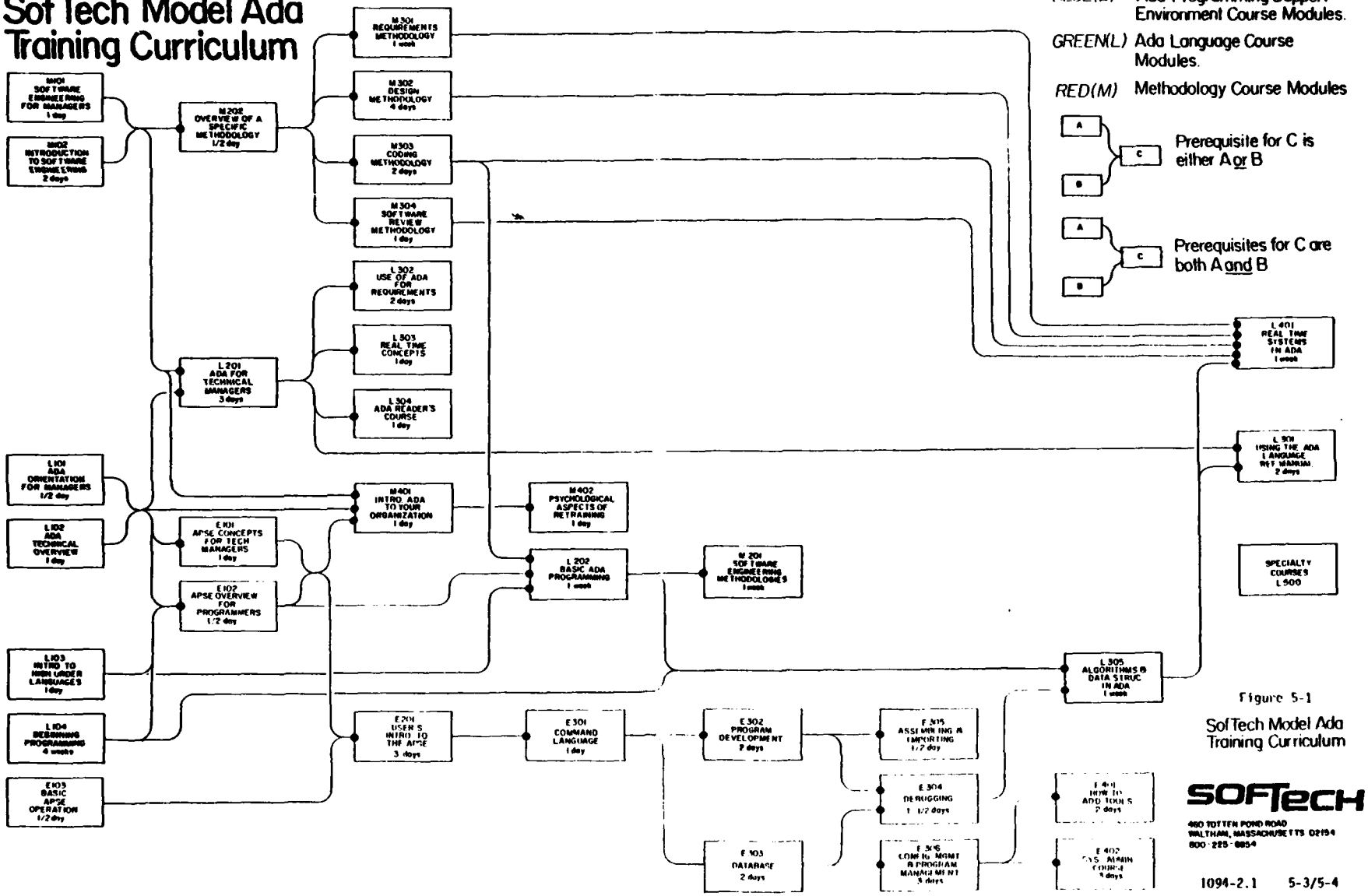


Figure 5-1

SofTech Model Ada Training Curriculum

SOFTECH
460 TOTTEN POND ROAD
WALTHAM, MASSACHUSETTS 02154
800-225-8854

MODULE DESCRIPTION

NAME: APSE Concepts for Technical Managers

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 101

DURATION: 1 day

PREREQUISITE:
Ada Orientation for Managers (L 101) or
Ada Technical Overview (L 102)

OBJECTIVE:
To provide managers with a technical overview of the APSE, emphasizing how it can be used to support the software life cycle.

SYLLABUS:
- Purpose and general description of the APSE
- APSE support for each life cycle phase
- Catalog of APSE tools
- Portable tools
- APSE support for configuration management
- APSE support for project management
- Concept of adding tools

RECOMMENDED FOR:
Senior Engineering Manager
Project/Task Leader
Support Manager
System Integration Manager/Research Staff

NOTES:
Most thorough of the three APSE introductory courses

MODULE DESCRIPTION

NAME: APSE Overview for Programmers

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 102

DURATION: 1/2 day

PREREQUISITE:

Ada Orientation for Managers (L 101) or
Ada Technical Overview (L 102) or
Introduction to HOL's (L 103) or
Beginning Programming with Ada (L 104)

OBJECTIVE:

To provide the programmer with an overview of the APSE, emphasizing how it supports the software life cycle.

SYLLABUS:

- Purpose and general description of the APSE
- APSE support for each life cycle phase
- Catalog of APSE tools
- Developing a program using the APSE
  - Editing
  - Compiling
  - Linking
  - Loading
  - Debugging
- Configuration management
- Editor

RECOMMENDED FOR:

Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Specialist
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Basic APSE Operation

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 103

DURATION: 1/2 day

PREREQUISITE:
None

OBJECTIVE:
To familiarize the student with the APSE tools and the database.

SYLLABUS:
- Purpose and general description of the APSE
- Catalog of APSE tools
- Structure of the database
- Editor

RECOMMENDED FOR:
Project Administrative Manager
Configuration Management/Quality Assurance Engineer (general background)
Junior Staff Member/Technical Aide

MODULE DESCRIPTION

NAME: USER'S INTRODUCTION TO THE APSE

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 201

DURATION: 3 Days

PREREQUISITE:

APSE concepts for Technical Managers (E 101) or
APSE overview for Programmers (E 102) or
Basic APSE Operation (E 103)

OBJECTIVE:

To give the student basic knowledge of the APSE

SYLLABUS:

- Description of the database
- File manipulation (e.g., renaming, printing)
- Logging in
- Command language overview
- Description and demonstration of each tool
- Editing

RECOMMENDED FOR:

Senior Engineering Manager
Project Administrative Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Specialist
Junior Staff Member/Technical Aide
Support Manager
System Integration Manager/Research Staff
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Command Language

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 301

DURATION: 1 day

PREREQUISITE:
User's Introduction to the APSE (E 201)

OBJECTIVE:
To teach the APSE command language.

SYLLABUS:
- Command language statements
- Substitutors
- I/O redirection
- Command procedures
- Background execution
- Interrupting program execution

RECOMMENDED FOR:
Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Junior Staff Member/Technical Aide
Support Manager
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Program Development

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 302

DURATION: 2 days

PREREQUISITE:
Command Language (E 301)

OBJECTIVE:
To teach program development (e.g., compiling, linking).

SYLLABUS:
- Separate compilation
- Program libraries
- Compiler
- Compilation order
- Linker
- Exporter
- Loading and executing a program
- Simple debugging
- Distributed processing

RECOMMENDED FOR:
Senior Engineering Manager
Support Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Junior Staff Member/Technical Aide
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Database

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 303

DURATION: 2 days

PREREQUISITE:
Command Language (E 301)

OBJECTIVE:
To teach the APSE database.

SYLLABUS:
- Files and file manipulation
- Path names
- Revisions
- Directories
- Variations
- Attributes
- Associations
- Access control
- Node sharing

RECOMMENDED FOR:
Senior Engineering Manger
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Support Manager
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Debugging

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 304

DURATION: 1 1/2 days

PREREQUISITE:

Program Development (E 302)
Database (E 303)

OBJECTIVE:

To teach the student to debug programs.

SYLLABUS:

- Debugger
- Frequency analyzer
- Timing analyzer
- Example of a debugging session
- Debugging on separate targets

RECOMMENDED FOR:

Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Assembling and Importing

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 305

DURATION: 1/2 day

PREREQUISITE:
Program Development (E 302)

OBJECTIVE:
To teach how to assemble and import subprograms.

SYLLABUS:
- Assembly language
- Directives
- Importer

RECOMMENDED FOR:
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Configuration Management and Program Management

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 306

DURATION: 3 days

PREREQUISITE:
Database (E 303)

OBJECTIVE:
To teach the concepts of configuration management and program management.

SYLLABUS:
- Strategies for configuration management
- APSE features which support configuration management
- Configuration management standards
- Strategies for program management
- APSE features which support program management
- Development of tools which automate configuration management and program management

RECOMMENDED FOR:
Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Real-Time System Architect
Support Manager
System Integration Engineer

MODULE DESCRIPTION

NAME: How to Add Tools

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 401

DURATION: 2 days

PREREQUISITE:
Configuration Management and Program Management (E 306)

OBJECTIVE:
To teach the student how to add tools to the APSE.

SYLLABUS:
- File system
- Protections
- Command procedures
- KAPSE interfaces
- Examples of tools

RECOMMENDED FOR:
Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
System Integration Engineer

MODULE DESCRIPTION

NAME: System Administrator's Course

AREA: Ada Programming Support Environment (APSE)

NUMBER: E 402

DURATION: 3 days

PREREQUISITE:
Configuration Management and Program Management (E 306)

OBJECTIVE:
To train the System Administrator.

SYLLABUS:
- Tasks performed by the System Administrator
- System installation
- User authorization
- Backup operations
- Archiving
- System support services

RECOMMENDED FOR:
Person designated as the System Administrator (Support Manager)

NOTES:
Walkthrough of the System Administrator's Handbook and exercises

MODULE DESCRIPTION

NAME: Ada Orientation for Managers

AREA: Ada Language

NUMBER: L 101

DURATION: 1/2 day

PREREQUISITE:
None

OBJECTIVE:
To give managers an overview of the development and features of Ada.

SYLLABUS:
- History and development of Ada
- Objectives of Ada
- Introduction to capabilities of Ada
- Overcoming potential resistance when introducing Ada
- Available Ada products (software, training, etc.)
- Future DoD plans
- Ada-related activities
  - ACVC
  - Methodman
  - Educationman

RECOMMENDED FOR:
Senior Engineering Manager
Project Administrative Manager
Support Manager
System Integration Manager/Research Staff

MODULE DESCRIPTION

NAME: Ada Technical Overview

AREA: Ada Language

NUMBER: L 102

DURATION: 1 day

PREREQUISITE:
None

OBJECTIVE:
To give the student an overview of the development and features of Ada.

SYLLABUS:
- History and development of Ada
- Objectives of Ada
- Catalog of features of Ada

RECOMMENDED FOR:
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Real-Time System Architect
Specialist
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Introduction to Higher Order Languages

AREA: Ada Language

NUMBER: L 103

DURATION: 1 day

PREREQUISITE:
None

OBJECTIVE:
To provide assembly language programmers with an understanding of higher order languages.

SYLLABUS:
- Advantages of higher order languages
- Translating higher order language programs into executable code
- Data types
- Control structures
- Expressions

RECOMMENDED FOR:
Configuration Management/Quality Assurance Engineer (general background)
Design Consultant
Programmer
Software Designer
Real-Time System Architect
Specialist
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Beginning Programming

AREA: Ada Language

NUMBER: L 104

DURATION: 4 weeks

PREREQUISITE:
None

OBJECTIVE:
Introduce the student to programming and teach him to write basic Ada programs.

SYLLABUS:
- How computers operate
- Software development environments
- Programming concepts
- Software design concepts
- Software life cycle
- Basic Ada programming

RECOMMENDED FOR:
Junior Staff Member/Technical Aide

NOTES:
For people who have never programmed, this course will contain extensive exercises and examples. This module could be an "after-hours" course.

MODULE DESCRIPTION

NAME: Ada for Technical Managers

AREA: Ada Language

NUMBER: L 201

DURATION: 3 days

PREREQUISITE:

Ada Orientation for Managers (L 101) or
Ada Technical Overview (L 102)
Software Engineering for Managers (M 101) or
Introduction to Software Engineering (M 102)

OBJECTIVE:

To teach students how to develop and recognize a high quality software design in Ada.

SYLLABUS:

- Using Ada features in software design
  - Packages
  - Private types
  - Generics
- Designing for reusability and portability
- Characteristics of a good Ada design

RECOMMENDED FOR:

Senior Engineering Manager
Project/Task Leader
Design Consultant
Support Manager
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
System Integration Manager/Research Staff
System Integration Senior Technical Staff
System Integration Engineer

NOTES:

For managers who do not design software, but direct those doing the design; the manager must be able to recognize a good software design. Have a sizable example to walk through. Pascal or JOVIAL background would make course shorter.

MODULE DESCRIPTION

NAME: Basic Ada Programming

AREA: Ada Language

NUMBER: L 202

DURATION: 1 week

PREREQUISITE:
APSE Overview for Programmers (E 102)
Introduction to Higher Order Languages (L 103)
Coding Methodology (M 303)

OBJECTIVE:
To teach the student to write basic Ada programs.

SYLLABUS:
- Overview of the Ada language
- Scalar data types
- Control structures
- Arrays and simple record types
- Subprograms
- Subprogram specifications and bodies
- Exceptions
- Concepts of packages and generics

RECOMMENDED FOR:
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

NOTES:
Provides the skills necessary to implement individual subprograms. Must teach how and when to use library units (subprograms, packages, generics). Provides the minimum knowledge needed to contribute, in a non-design role, to an Ada project.

MODULE DESCRIPTION

NAME: Using the Ada Language Reference Manual

AREA: Ada Language

NUMBER: L 301

DURATION: 2 days

PREREQUISITE:

Ada for Technical Managers (L 201)
Algorithms and Data Structures in Ada (L 305)

OBJECTIVE:

To use the Ada Language Reference Manual to resolve language interpretation issues.

SYLLABUS:

- Structure of the ALRM
- Syntactic notation
- Using the ALRM to understand semantic details
- Walkthrough of selected chapters of the ALRM

RECOMMENDED FOR:

Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)

NOTES:

The implementor's guide could be used as a textbook.

MODULE DESCRIPTION

NAME: Use of Ada for Requirements

AREA: Ada Language

NUMBER: L 302

DURATION: 2 days

PREREQUISITE:
Ada for Technical Managers (L 201)

OBJECTIVE:
To teach students how to express system requirements in Ada.

SYLLABUS:
- Objectives of a requirements specification
- Use of Ada constructs at the requirements level
- Expressing constraints and timing requirements in Ada
- Specifying reusable software packages

RECOMMENDED FOR:
System Integration Manager/Research Staff
System Integration Senior Technical Staff
System Integration Engineer

NOTES:
This module could be part of a series of special topics.

MODULE DESCRIPTION

NAME: Real-Time Concepts

AREA: Ada Language

NUMBER: L 303

DURATION: 1 day

PREREQUISITE:
Ada for Technical Managers (L 201)

OBJECTIVE:
To teach managers approaches to real-time programming.

SYLLABUS:
- Synchronous/asynchronous system design
- Ada tasking
- Interrupt handling
- Queue management
- Role of a run-time system

RECOMMENDED FOR:
Project/Task Leader

NOTES:
For project leaders who need to understand designs and settle disputes. Not how to do real-time programming; rather, understanding issues about and approaches to real-time programming. Few examples; not many exercises.

MODULE DESCRIPTION

NAME: Ada Reader's Course

AREA: Ada Language

NUMBER: L 304

DURATION: 1 day

PREREQUISITE:
Ada for Technical Managers (L 201)

OBJECTIVE:
To develop the ability to read an Ada program for its structure.

SYLLABUS:
- Ada program structure
  - Subprograms
  - Packages
  - Subunits
  - Stubs
  - Tasks
- Reading Ada program designs and code
- Recognizing poor design choices

RECOMMENDED FOR:
Configuration Management/Quality Assurance Engineer (general background)
Senior Engineering Manager
Project/Task Leader
System Integration Manager/Research Staff

NOTES:
Student is not responsible for determining whether designs or code are correct. Checklist approach.

MODULE DESCRIPTION

NAME: Algorithms and Data Structures in Ada

AREA: Ada Language

NUMBER: L 305

DURATION: 1 week

PREREQUISITE:

Basic Ada Programming (L 202) or
Beginning Programming with Ada (L 104)
Debugging (E 304)

OBJECTIVE:

To teach the Ada features used in programming.

SYLLABUS:

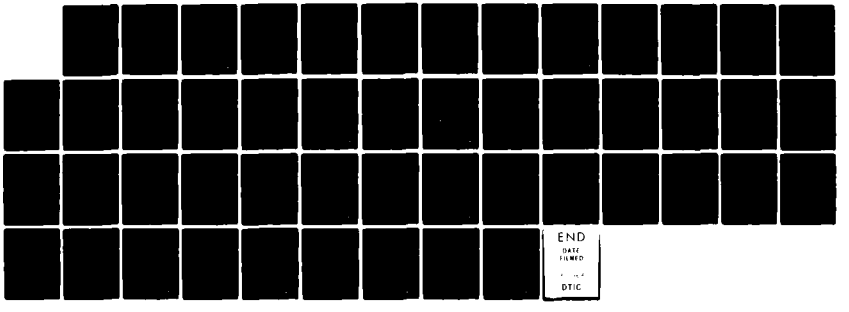
- Discriminated record types
- Record variants
- Access types
- Derived types
- Packages
- Private types
- Generics
- Overloading
- Practice with common algorithms and data structures
- Low-level machine-dependent features
  - Representation specifications
  - Unchecked conversions
- Introduction to tasking

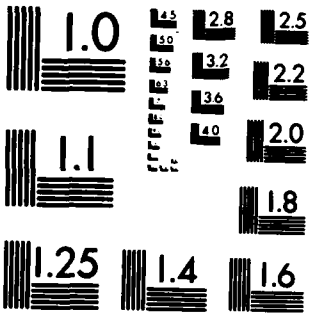
RECOMMENDED FOR:

Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

AD-A124 998 ADA* SOFTWARE DESIGN METHODS FORMULATION(U) SOFTECH INC 3/3
WALTHAM MA OCT 82 DAAK80-80-C-0187

UNCLASSIFIED F/G 9/2 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

MODULE DESCRIPTION

NAME: Real-Time Systems in Ada

AREA: Ada Language

NUMBER: L 401

DURATION: 1 week

PREREQUISITE:

Algorithms and Data Structures (L 305)
Requirements Methodology (M 301)
Design Methodology (M 302)
Coding Methodology (M 303)
Software Review Methodology (M 304)

OBJECTIVE:

To teach the Ada features used to program real-time systems.

SYLLABUS:

- Synchronous/asynchronous system design
- Tasking
- Task types
- Interrupt handling
- Queue management
- Role of a run-time system
- Resource scheduling
- Timing and schedule
- Low-level I/O

RECOMMENDED FOR:

Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Specialy Courses

AREA: Ada Language

NUMBER: L 500

DURATION: Varying according to area

PREQUISITE:
As appropriate.

OBJECTIVE:
The objective of the specialty courses is to offer indepth training in various "specialty" areas in programming (i.e., numerical algorithms, graphics, man/machine interface, etc.).

SYLLABUS:
- Varying according to area

RECOMMENDED FOR:
Specialist

MODULE DESCRIPTION

NAME: Software Engineering for Managers

AREA: Methodology

NUMBER: M 101

DURATION: 1 day

PREREQUISITE:
None

OBJECTIVE:
To teach managers modern software engineering concepts.

SYLLABUS:
- Software life cycle
- The importance of specifying requirements
- Top-down and bottom-up development
- Software documentation
- Role of quality assurance and configuration management
- Approaches to software testing and integration

RECOMMENDED FOR:
Senior Engineering Manager
Project Administrative Manager
Project/Task Leader
Support Manager
System Integration Manager/Research Staff

NOTES:
Numbers and charts to show the measurable value of techniques.

MODULE DESCRIPTION

NAME: Introduction to Software Engineering

AREA: Methodology

NUMBER: M 102

DURATION: 2 days

PREREQUISITE:
None

OBJECTIVE:
To teach engineers software engineering concepts.

SYLLABUS:
- Software life cycle
- Top-down and bottom-up development
- Introduction to methodologies
  - Structured design
  - HIPO
  - Data flow diagrams
- Software documentation
- Approaches to software testing

RECOMMENDED FOR:
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Software Engineering Methodologies

AREA: Methodology

NUMBER: M 201

DURATION: 1 week

PREREQUISITE:
Basic Ada Programming (L 202)

OBJECTIVE:
To teach a thorough understanding of software methodologies and how they may be used with Ada. To provide a basis for selecting a corporate set of methodologies.

SYLLABUS:
- Alternative models of the software life cycle
- Objective of each phase of the life cycle
- Several software methodologies
  - Structured analysis
  - Structured design
  - HIPO
  - Entity diagrams
  - Object-oriented design
  - Structured programming
  - SADT
  - Software review techniques
- How these methodologies may be used with Ada

RECOMMENDED FOR:
Design Consultant
System Integration Senior Technical Staff

MODULE DESCRIPTION

NAME: Overview of a Specific Methodology

AREA: Methodology

NUMBER: M 202

DURATION: 1/2 day

PREREQUISITE:

Software Engineering for Managers (M 101) or
Introduction to Software Engineering (M 102)

OBJECTIVE:

To provide an overview of the methodologies selected by an organization.

SYLLABUS:

- Life cycle phases
- Overview of the techniques used during each stage of the life cycle
- Documenting the products of each stage of the life cycle

RECOMMENDED FOR:

Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (general and in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
Support Manager
System Integration Manager/Research Staff
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Requirements Methodology

AREA: Methodology

NUMBER: M 301

DURATION: 1 week

PREREQUISITE:
Overview of a Specific Methodology (M 202)

OBJECTIVE:
To teach students to define requirements using a specific methodology.

SYLLABUS:
- Approaches to defining system requirements
- Using the requirements methodology
- Producing the requirements definition documentation

RECOMMENDED FOR:
Senior Engineering Manager
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Design Methodology

AREA: Methodology

NUMBER: M 302

DURATION: 4 days

PREREQUISITE:
Overview of a Specific Methodology (M 202)

OBJECTIVE:
To teach students to design using a specific methodology.

SYLLABUS:
- Approaches to software design
  - Top-down design
  - Levels of abstraction
  - Bottom-up design
  - Coupling and cohesion
  - Information hiding
- Program design language
- Using the design methodology
- Producing design documentation

RECOMMENDED FOR:
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Coding Methodology

AREA: Methodology

NUMBER: M 303

DURATION: 2 days

PREREQUISITE:
Overview of a Specific Methodology (M 202)

OBJECTIVE:
To teach coding and documentation conventions.

SYLLABUS:
- Structured programming concepts
- Basic control structures
- The structure theorem
- In-line documentation requirements
- Producing the required documentation
- Programming style
- Exercises

RECOMMENDED FOR:
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Software Review Methodology

AREA: Methodology

NUMBER: M 304

DURATION: 1 day

PREREQUISITE:
Overview of a Specific Methodology (M 202)

OBJECTIVE:
To teach software review techniques.

SYLLABUS:
- Individual software review
- Code reading (peer review)
- Structured walkthroughs
- Checklists for reviews

RECOMMENDED FOR:
Project/Task Leader
Design Consultant
Configuration Management/Quality Assurance Engineer (in-depth technical background)
Programmer
Software Designer
Real-Time System Architect
System Integration Senior Technical Staff
System Integration Engineer

MODULE DESCRIPTION

NAME: Introducing Ada to Your Organization

AREA: Methodology

NUMBER: M 401

DURATION: 1 day

PREREQUISITE:

APSE Concepts for Technical Managers (E 101) or
APSE Overview for Programmers (E 102)
Software Engineering for Managers (M 101) or
Introduction to Software Engineering (M 102)
Ada Orientation for Managers (L 101) or
Ada Technical Overview (L 102)

OBJECTIVE:

To teach how the recommended Ada curriculum might be introduced into an organization.

SYLLABUS:

- Recommended Ada curriculum
- Tailoring the curriculum to an organization's needs
- Available training resources
- Guidelines for scheduling courses
- Recommendations for the size and the composition of classes

RECOMMENDED FOR:

Design Consultant
Senior Engineering Manager
System Integration Senior Technical Staff

MODULE DESCRIPTION

NAME: Psychological Aspects of Training

AREA: Methodology

NUMBER: M 402

DURATION: 1 day

PREREQUISITE:
Introducing Ada to Your Organization (M 101)

OBJECTIVE:
To teach techniques for overcoming employees' resistance to change.

SYLLABUS:
- Reasons for resistance
  - Fear of obsolescence
  - The "larger scheme"
- Patterns of behavior which indicate resistance
- Techniques for overcoming resistance

RECOMMENDED FOR:
Design Consultant
System Integration Senior Technical Staff

NOTES:
A workshop rather than a lecture.

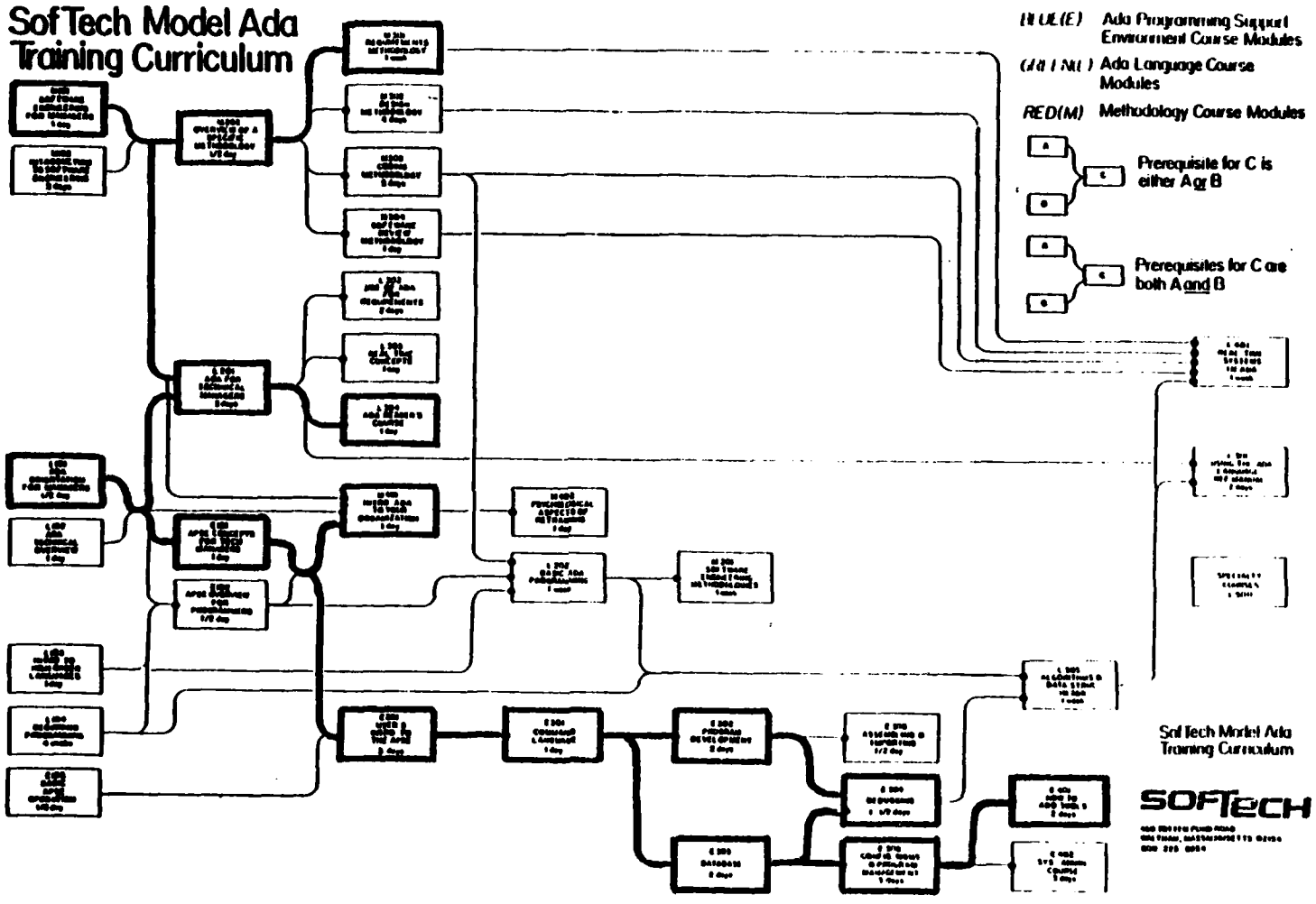
5.3 Recommended Curriculum by Job Category

The following pages represent, in graphical form, the curriculum recommended by SofTech for each of the generic job categories identified in Sections 3 and 4. As described in Section 5.1 each chart is to be read from left to right and the flow is to be along straight or curved lines - right angles are not intended to be paths. The recommended curriculum for the job category being described is highlighted by a thick dark line outlining the module names and the suggested paths. It should be stressed that these represent recommendations only, and that individual training needs will undoubtedly vary by background.

RECOMMENDED CIRRICULUM:

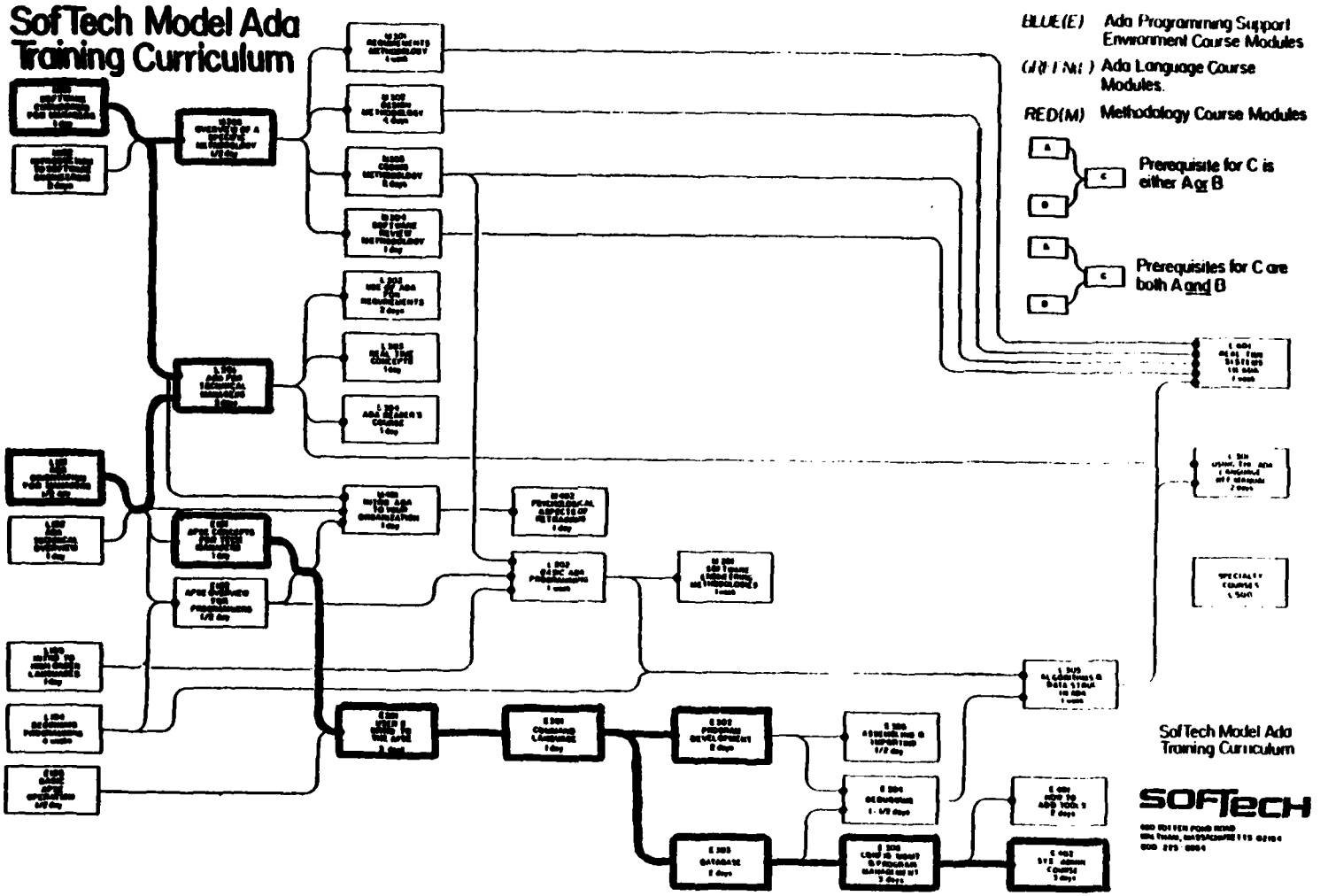
PROJECT ADMINISTRATIVE MANAGER

SOFTECH



RECOMMENDED CURRICULUM:

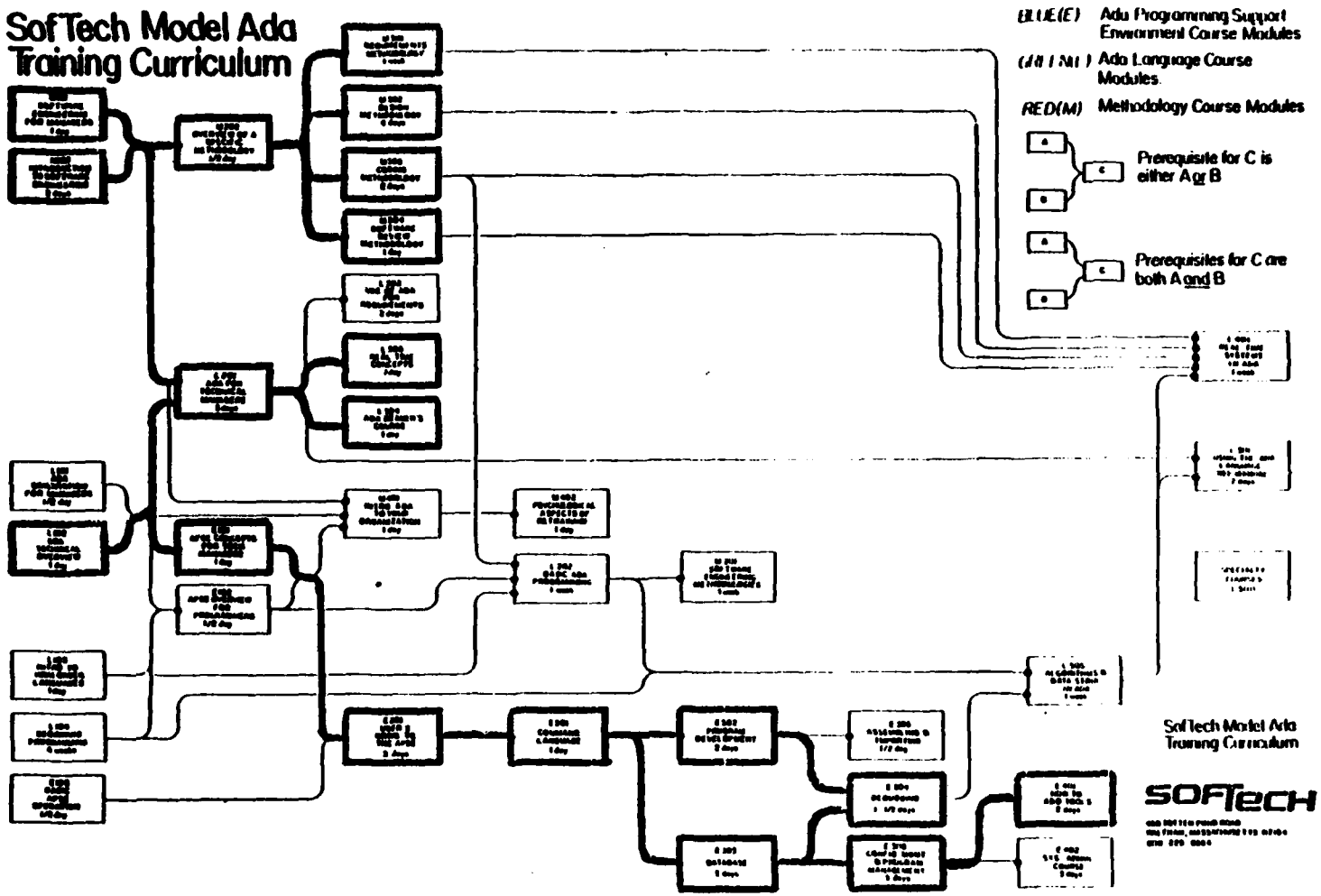
SENIOR ENGINEERING MANAGER



RECOMMENDED CIRRICULUM:

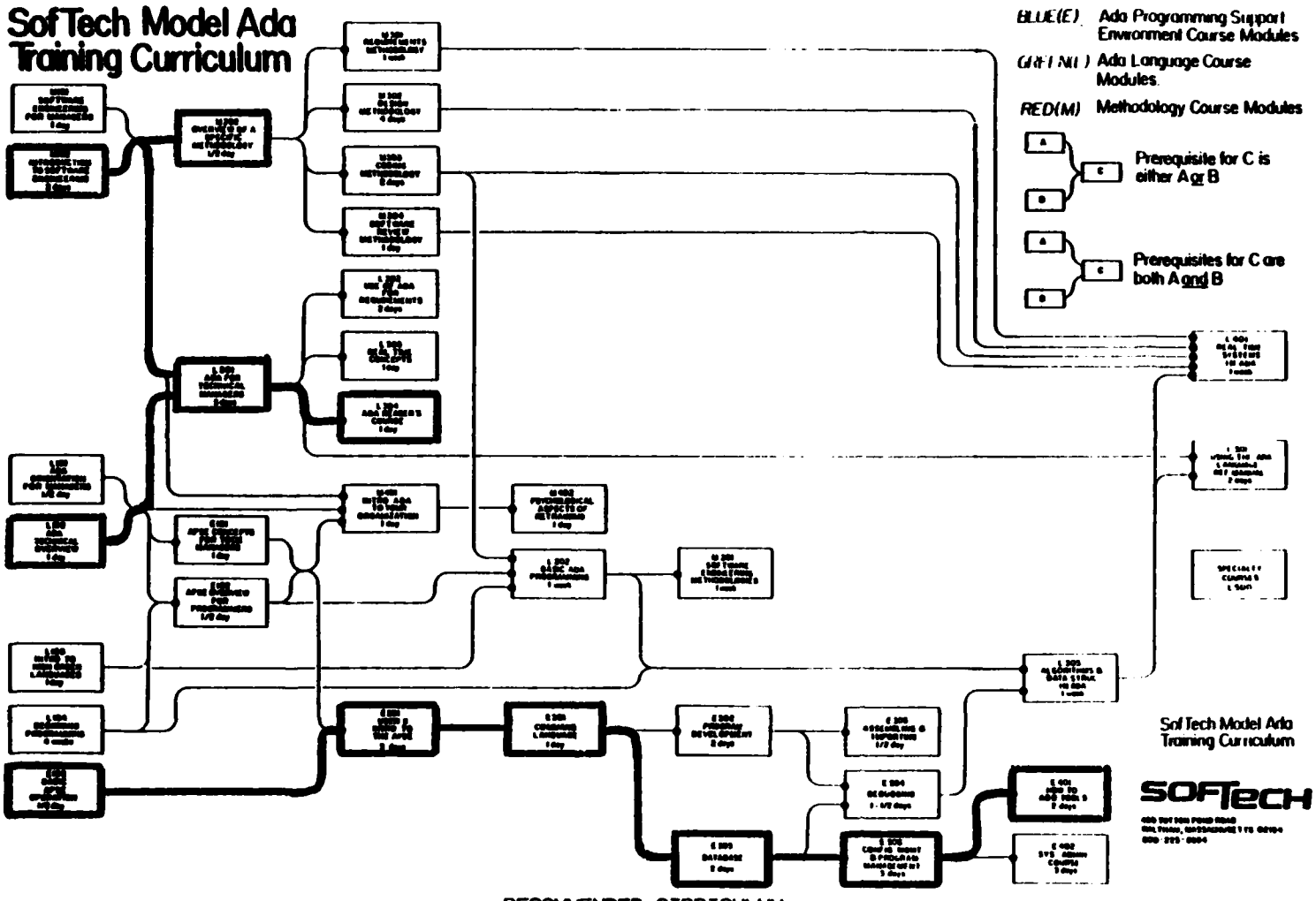
SUPPORT MANAGER

SOFTECH



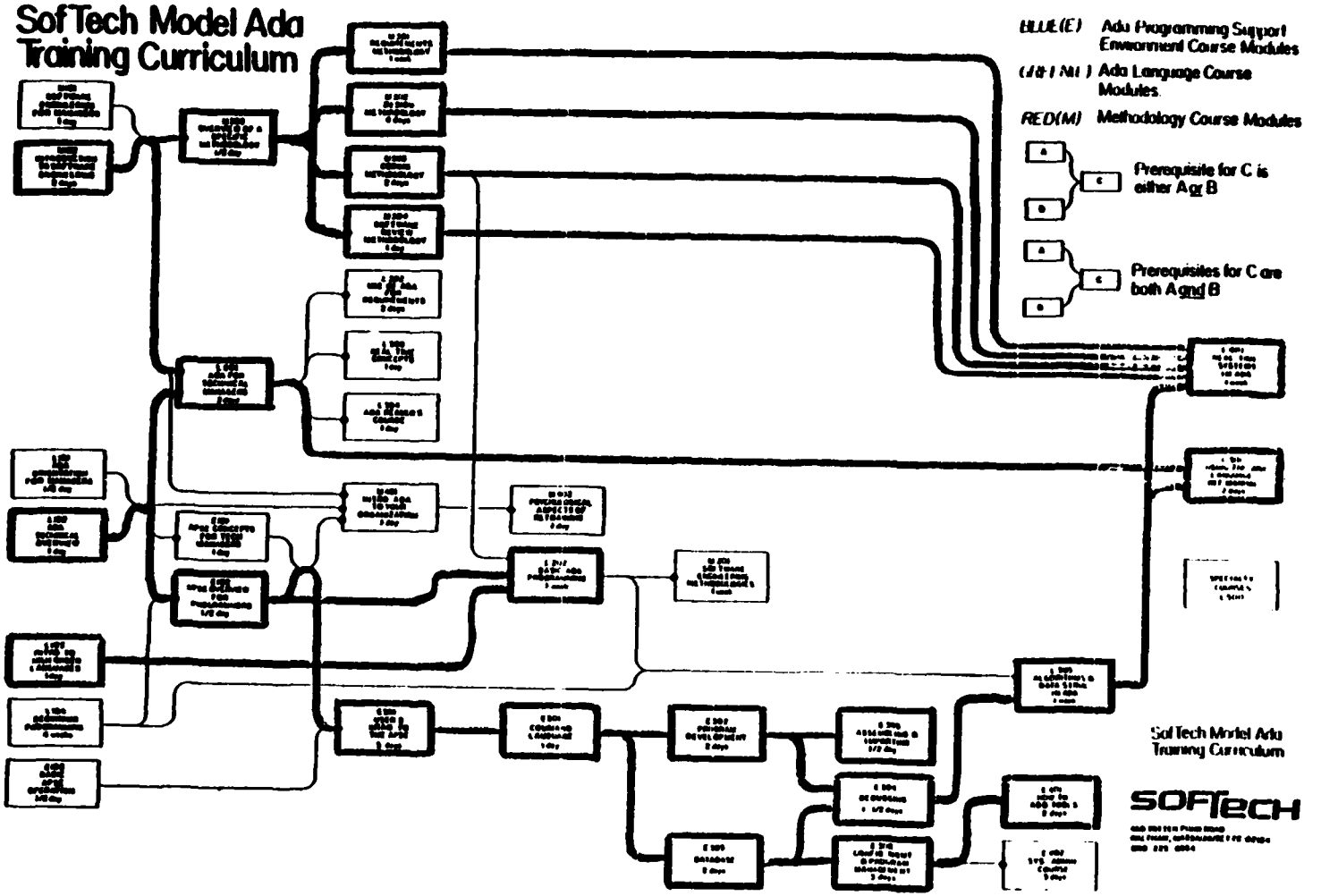
RECOMMENDED CURRICULUM:

PROJECT/TASK LEADER



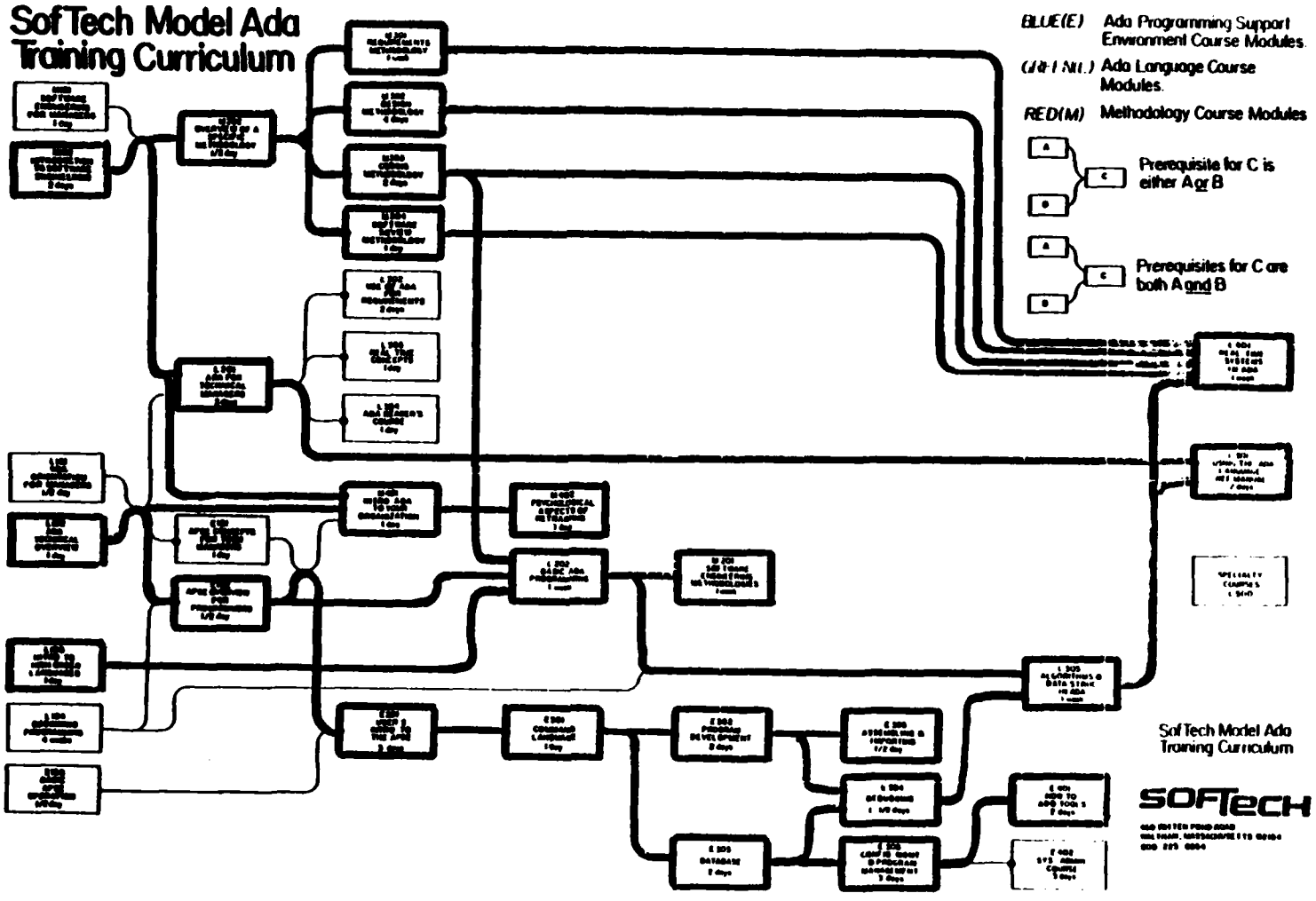
RECOMMENDED CIRRICULUM:

CONFIGURATION MANAGEMENT/QUALITY ASSURANCE
GENERAL BACKGROUND



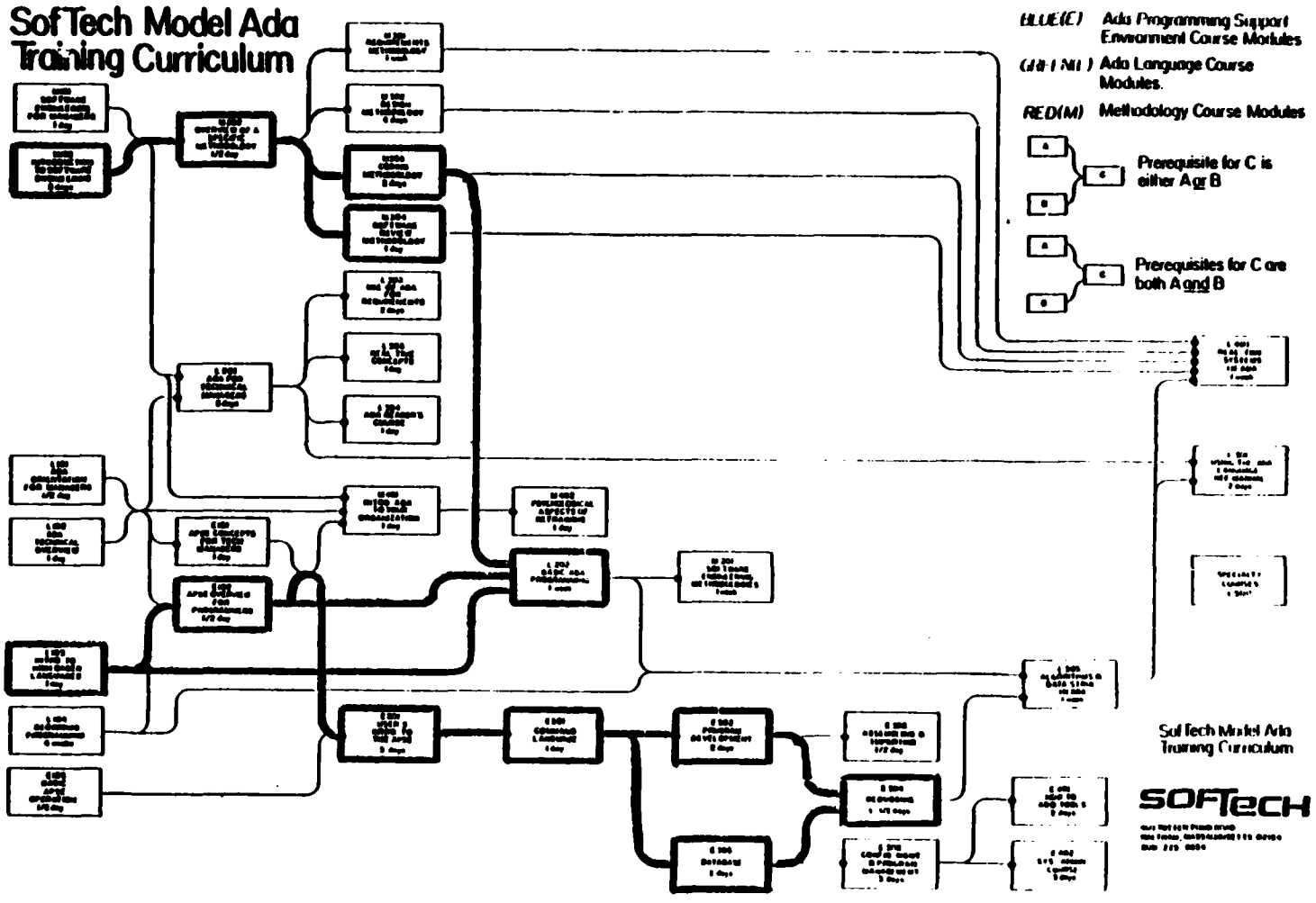
RECOMMENDED CURRICULUM:

CONFIGURATION MANAGEMENT/QUALITY ASSURANCE
IN-DEPTH TECHNICAL BACKGROUND



RECOMMENDED CURRICULUM:

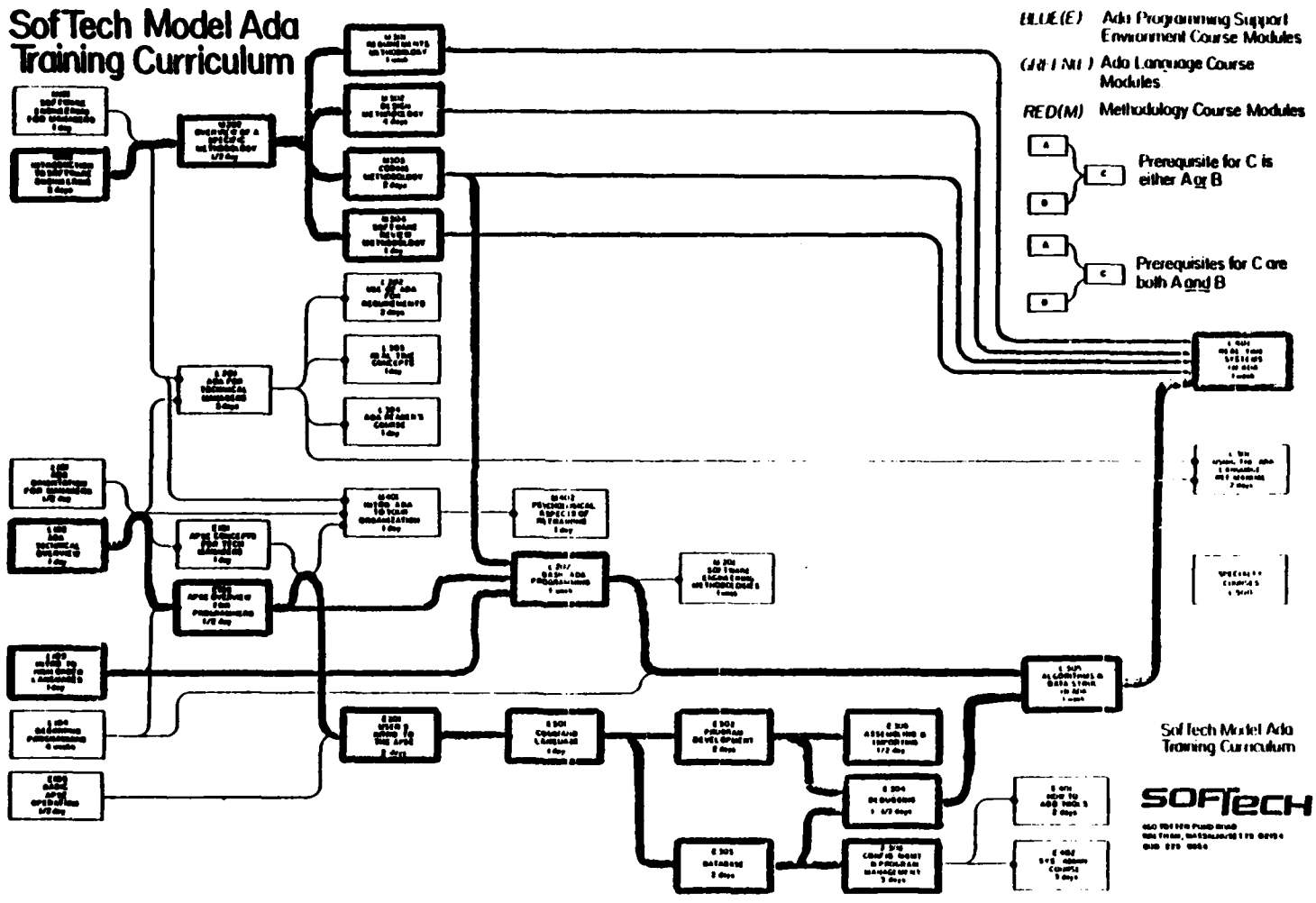
DESIGN CONSULTANT



RECOMMENDED CURRICULUM:

PROGRAMMER

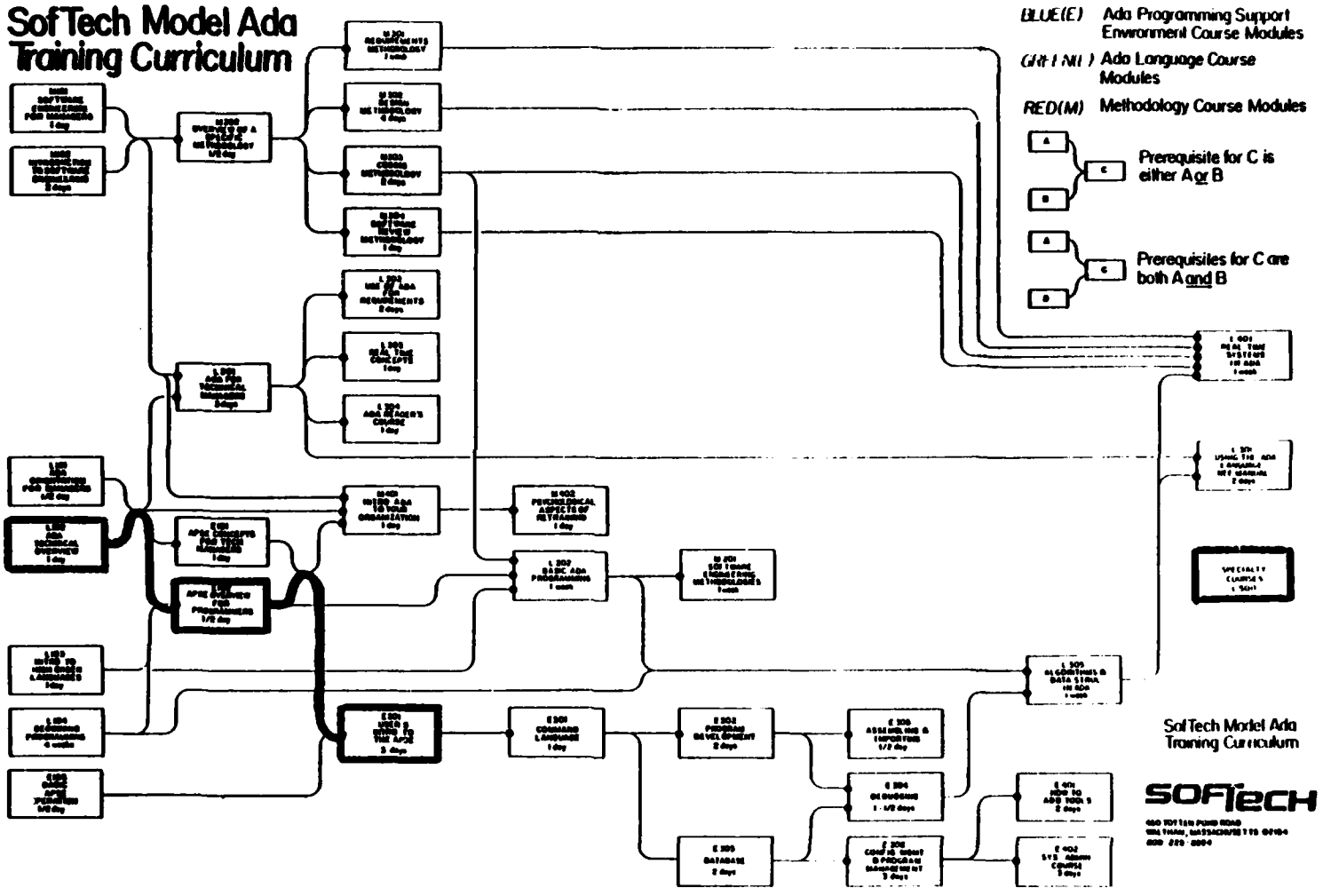
BLUE(E) Ada Programming Support Environment Course Modules
GREEN(L) Ada Language Course Modules.
RED(M) Methodology Course Modules
Prerequisite for C is either A or B
Prerequisites for C are both A and B
SofTech Model Ada Training Curriculum
SofTech Model Ada Training Curriculum
SOFTECH
RECOMMENDED CURRICULUM:
SOFTWARE DESIGNER



RECOMMENDED CURRICULUM:

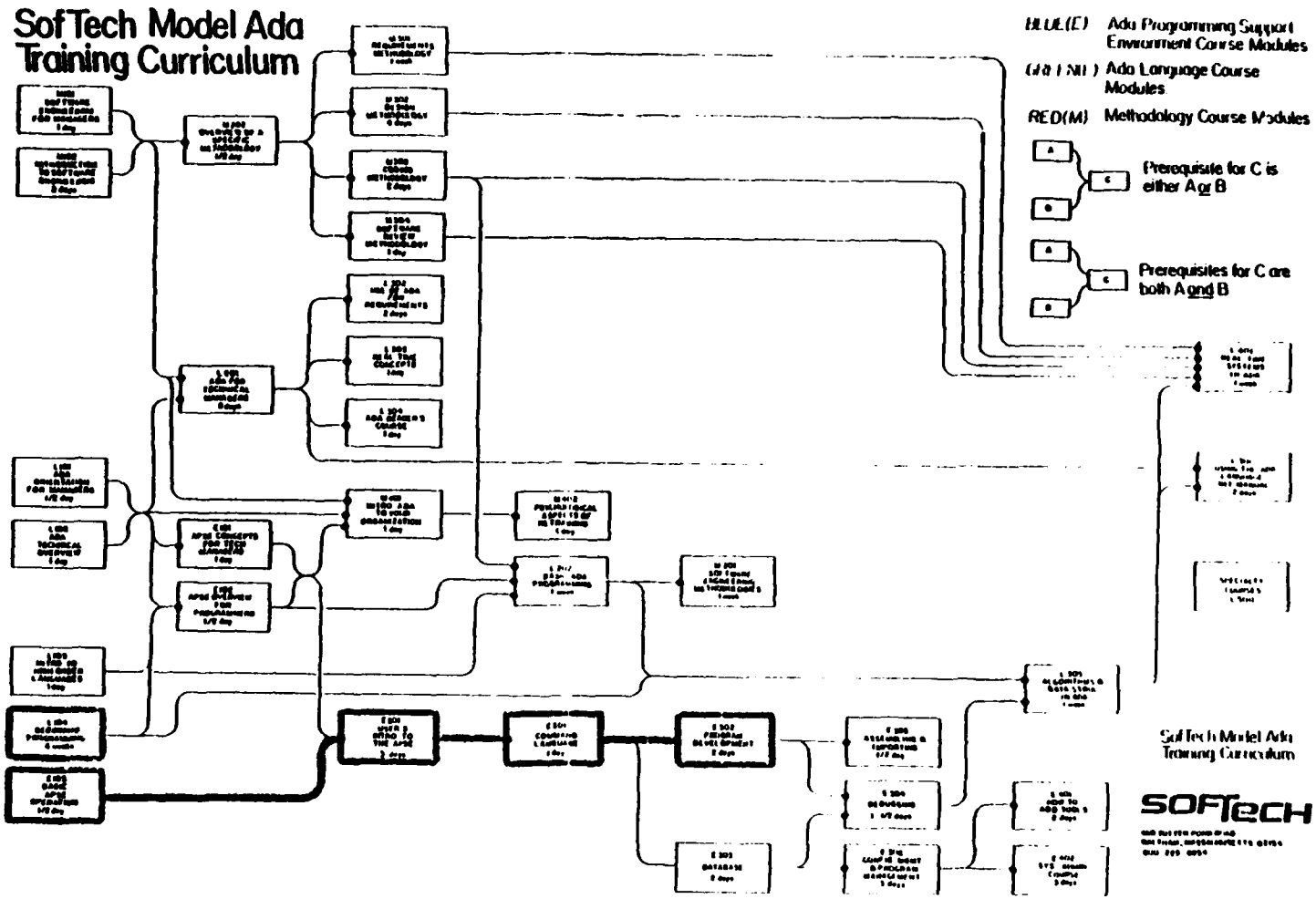
REAL-TIME SYSTEM ARCHITECT

SofTech Model Ada Training Curriculum



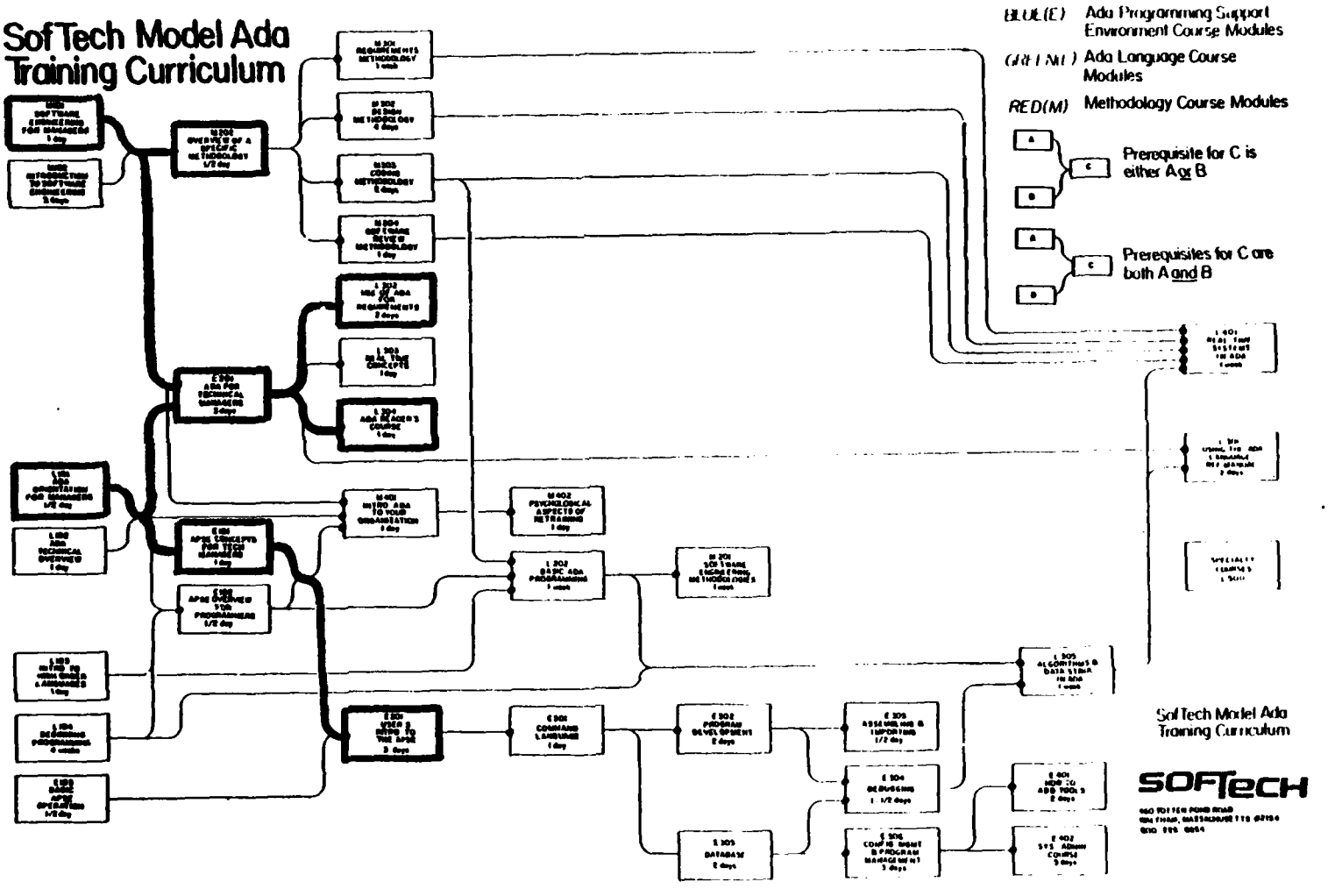
RECOMMENDED CURRICULUM:

SPECIALIST



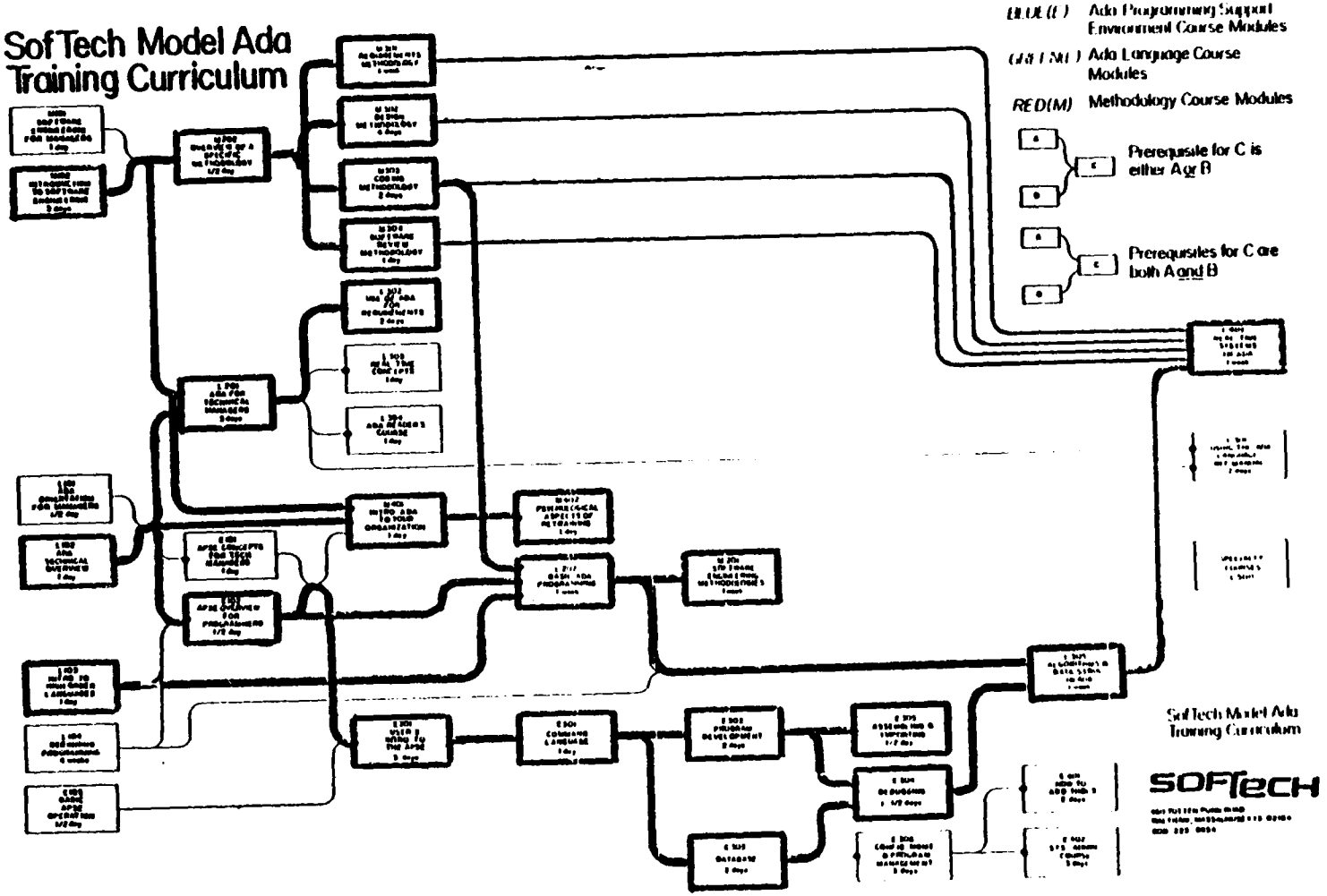
RECOMMENDED CURRICULUM:

JUNIOR STAFF MEMBER/TECHNICAL AIDE



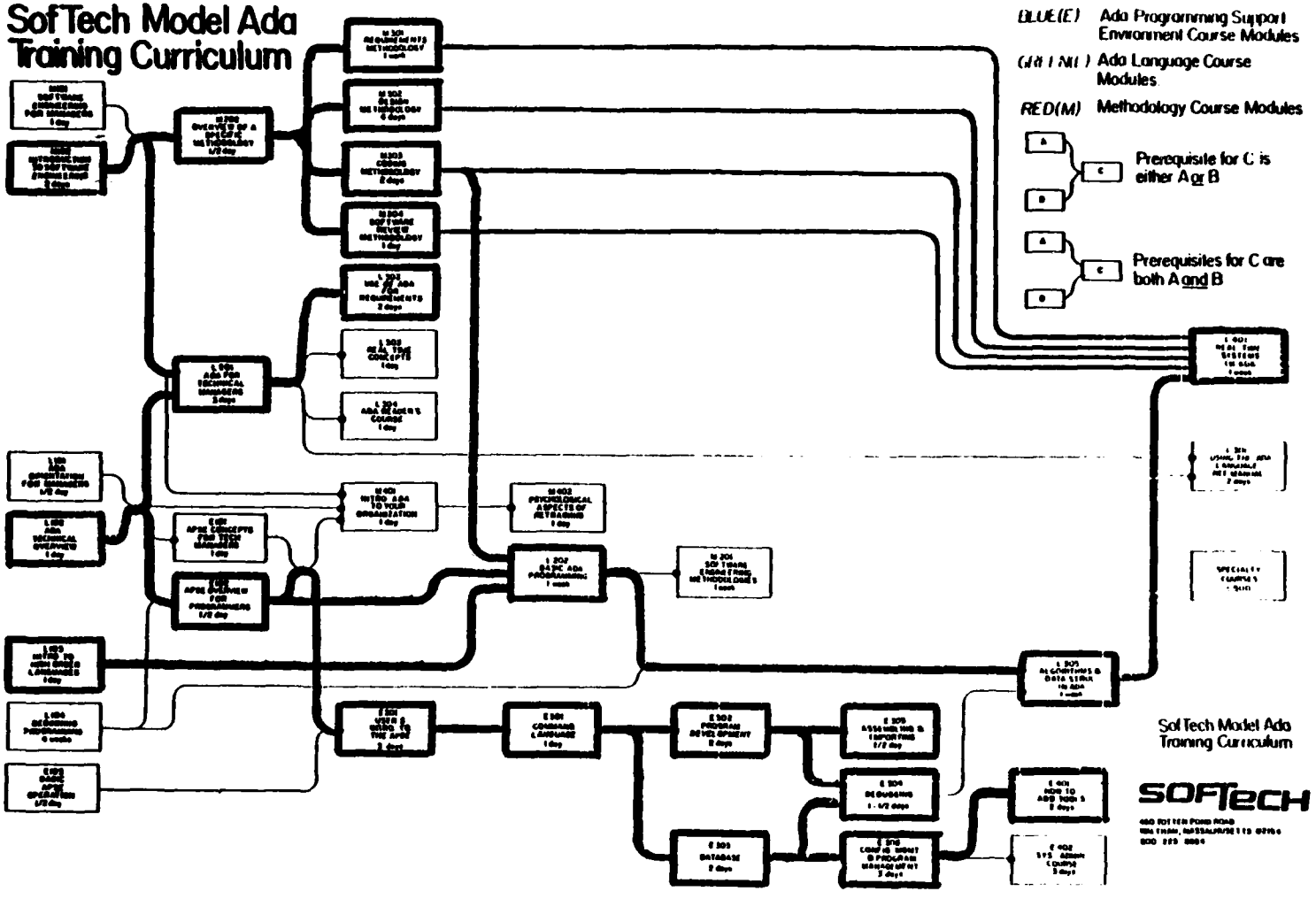
RECOMMENDED CURRICULUM:

SYSTEM INTEGRATION MANAGER/RESEARCH STAFF



RECOMMENDED CURRICULUM:

SYSTEM INTEGRATION SENIOR TECHNICAL STAFF



RECOMMENDED CURRICULUM:

SYSTEM INTEGRATION ENGINEER

5.4 Cross-reference Tables

This section provides tables cross-referencing curriculum modules with both job categories and case studies.

Figure 5-2 shows APSE course modules cross-referenced with the generic job categories, Figure 5-3 shows the Ada language course modules cross-referenced with the generic job categories, and Figure 5-4 shows the Methdology course modules cross-referenced with the generic job categories.

Figure 5-5 cross-references the Ada language course modules with the case studies completed under the present contract and Figure 5-6 cross-references the methodology course modules with completed case studies.

SofTech has identified areas of research for future case studies (See Appendix A of Case Studies Report). These future case studies are cross-referenced by Ada Language Course modules in Figure 5-7 and by Methodology Course modules in Figure 5-8.

Ada Programming Support Environment Course Modules and Job Categories	ASPE Concepts for Technical Managers (E101)	APSE Overview for Programmers (E102)	Basic APSE Operation (E103)	User's Introduction to the APSE (E201)	Command Language (E301)	Program Development (E302)	Database (E303)	Debugging (E304)	Assembling and Importing (E305)	Configuration Management and Program Management (E306)	How to Add Tools (E401)	System Administrator's Course (E402)
Project Administrative Manager			•	•								
Senior Engineering Manager	•			•	•	•	•	•		•	•	
Support Manager	•			•	•	•	•			•		•
Project/Task Leader	•			•	•	•	•	•		•	•	
Configuration Management/ Quality Assurance Engineer (General Background) (In-depth Technical Background)		•	•	• •	• •	•	• •	•	•	• •	• •	
Design Consultant		•		•	•	•	•	•	•	•	•	
Programmer		•		•	•	•	•	•				
Software Designer		•		•	•	•	•	•	•			
Real-Time System Architect		•		•	•	•	•	•	•	•		
Specialist		•		•								
Junior Staff Member/Technical Aide			•	•	•	•						
System Integration Manager/ Research Staff	•			•								
System Integration Senior Technical Staff		•		•	•	•	•	•	•			
System Integration Engineer		•		•	•	•	•	•	•	•	•	
Specialist	Varying According to Specialty											

Figure 5-2. APSE Course Modules Cross-reference with Job Categories

Ada Language Course Modules and Job Categories	Ada Orientation for Managers (L101)	Ada Technical Overview (L102)	Introduction to High Order Languages (L103)	Beginning Programming with Ada (L104)	Ada for Technical Managers (L201)	Basic Ada Programming (L202)	Using the Ada Language Reference Manual (L301)	Use of Ada for Requirements (L302)	Real-Time Concepts (L303)	Ada Reader's Course (L304)	Algorithms and Data Structures in Ada (L305)	Real-Time Systems in Ada (L401)	Specialty Courses
Project Administrative Manager	•												
Senior Engineering Manager	•				•					•			
Support Manager	•				•								
Project/Task Leader		•			•				•	•			
Configuration Management/ Quality Assurance Engineer (General Background) (In-depth Technical Background)		• •	•		• •	•	•			•	•	•	
Design Consultant		•	•		•	•	•				•	•	
Programmer			•			•							
Software Designer			•			•					•		
Real-Time System Architect		•	•			•					•	•	
Specialist		•	•										•
Junior Staff Member/Technical Aide				•									
System Integration Manager/ Research Staff	•				•			•		•			
System Integration Senior Technical Staff		•	•		•	•		•			•	•	
System Integration Engineer		•	•		•	•		•			•	•	
Specialist	Varying According to Specialty												

Figure 5-3. Ada Language Course Modules Cross-referenced with Job Categories

Methodology Course Modules and Job Categories	Software Engineering for Managers (M101)	Introduction to Software Engineering (M102)	Software Engineering Methodologies (M201)	Overview of a Specific Methodology (M202)	Requirements Methodology (M301)	Design Methodology (M302)	Coding Methodology (M303)	Software Review Methodology (M304)	Introducing Ada to Your Organization (M401)	Psychological Aspects of Retraining (M402)
Project Administrative Manager	•									
Senior Engineering Manager	•			•	•				•	
Support Manager	•			•						
Project/Task Leader	•	•		•	•	•	•	•		
Configuration Management/ Quality Assurance Engineer (General) Background) (In-depth Technical Background)		• •		• •	•	•	•	•		
Design Consultant		•	•	•	•	•	•	•	•	•
Programmer		•		•			•	•		
Software Designer		•		•	•	•	•	•		
Real-Time System Architect		•		•	•	•	•	•		
Specialist										
Junior Staff Member/Technical Aide										
System Integration Manager/ Research Staff	•			•						
System Integration Senior Technical Staff		•	•	•	•	•	•	•	•	•
System Integration Engineer		•		•	•	•	•	•		
Specialty	Varying According to Specialty									

Figure 5-4. Methodology Course Modules Cross-referenced with Job Categories

Ada Language Course Modules and Completed Case Studies	Ada Orientation for Managers (L101)	Ada Technical Overview (L102)	Introduction to High Order Languages (L103)	Beginning Programming with Ada (L104)	Ada for Technical Managers (L201)	Basic Ada Programming (L202)	Using the Ada Language Reference Manual (L301)	Use of Ada for Requirements (L302)	Real-Time Concepts (L303)	Ada Reader's Course (L304)	Algorithms and Data Structures in Ada (L305)	Real-Time Systems in Ada (L401)	Specialty Courses
Power Failure Requirements								•					
Use of Types to Describe Hardware Interface Requirements					•			•					
Functional Description of an Air Defense System													
Task Structure for a Target Tracking System												•	
UART: Expressing Hardware Design in Ada								•	•		•	•	
Tasks and Structure Charts									•	•		•	
Use of Dependent Tasks												•	
Task Preemption												•	
Queues and Generics									•	•	•		
Stubbing and Readability										•	•		
Succintness of Range Syntax											•		
Low-Level I/O											•	•	
Rendezvous and Exit												•	
Array of Arrays						•					•		
Decoupling Partly Independent Activities												•	
Eliminating Goto's						•				•	•	•	

Figure 5-5. Ada Language Course Modules Cross-referenced with Completed Case Studies

Methodology Course Modules and Completed Case Studies	Software Engineering for Managers (M101)	Introduction to Software Engineering (M102)	Software Engineering Methodologies (M201)	Overview of a Specific Methodology (M202)	Requirements Methodology (M301)	Design Methodology (M302)	Coding Methodology (M303)	Software Review Methodology (M304)	Introducing Ada to Your Organization (M401)	Psychological Aspects of Retraining (M402)
Power Failure Requirements					•					
Use of Types to Describe Hardware Interface Requirements										
Functional Description of an Air Defense System			•		•					
Task Structure for Target Tracking System										
UART: Expressing Hardware Design in Ada										
Tasks and Structure Charts										
Use of Dependent Tasks										
Task Preemption										
Queues and Generics										
Stubbing and Readability										
Succintness of Range Syntax										
Memory-Mapped I/O in Ada										
Rendezvous and Exit										
Array of Arrays										
Decoupling Partly Independent Activities										
Eliminating Goto's										

Figure 5-6. Methodology Course Modules Cross-referenced with Completed Case Studies

Methodology Course Modules and Future Case Studies	Software Engineering for Managers (M101)	Introduction to Software Engineering (M102)	Software Engineering Methodologies (M201)	Overview of a Specific Methodology (M202)	Requirements Methodology (M301)	Design Methodology (M302)	Coding Methodology (M303)	Software Review Methodology (M304)	Introducing Ada to Your Organization (M401)	Psychological Aspects of Retraining (M402)
Expressing High-Level Performance Constraints					•					
Single-Threaded Protocols					•					
Steady-State Approach						•				
Guidelines in Designing One Task Per "what"						•				
Modeling Finite State Machine						•				
Methodological Issue: Multiple/Distributed Processing						•				
Generic Types - System Security and Testability										
Managing a Common Storage Pool										
Alternative Task Selection										
Use of Parameters at . Particular Level of Design										
Information Hiding - Data Abstraction						•				
Postponing Design Details						•				
Modeling Tasks with SADT, DFD's Structure Charts						•				
Guidelines on Packages						•				
Tutorial on Exceptions										
Handling Impossible States										
Dot Notation, Naming and Readability							•			
Standard Coding Style										
Avoiding Busy Waits in Select Statement										
Simple Coding Paradigms										
Reusable Software Modules										
Hardware Error Detection										
Customized Run.-Time Systems										
Parameters to Analyze Tasking Performance										
Analysis of User Requirements					•					

Figure 5-7. Ada Language Course Modules Cross-referenced with Future Case Studies

Ada Language Course Modules and Future Case Studies	Ada Orientation for Managers (L101)	Ada Technical Overview (L102)	Introduction to High Order Languages (L103)	Beginning Programming with Ada (L104)	Ada for Technical Managers (L201)	Basic Ada Programming (L202)	Using the Ada Language Reference Manual (L301)	Use of Ada for Requirements (L302)	Real-Time Concepts (L303)	Ada Reader's Course (L304)	Algorithms and Data Structures in Ada (L305)	Real-Time Systems in Ada (L401)	Specialty Courses
Expressing High-Level Performance Constraints								•					
Single-threaded Protocols								•	•	•			•
Steady-state Approach								•	•			•	
Guidelines in Designing One Task Per "what"					•				•			•	
Modeling Finite State Machine									•			•	
Methodological Issue: Multiple/Distributed Processing								•	•			•	•
Generic Types - System Security and Testability					•						•		
Managing a Common Storage Pool											•	•	•
Alternative Task Selection												•	
Use of Parameters at a Particular Level of Design											•		
Information Hiding - Data Abstraction											•		
Postponing Design Details					•					•		•	
Modeling Tasks with SADT, DFD's, Structure Charts									•				
Guidelines on Packages					•	•					•		
Tutorial on Exceptions						•					•	•	
Handling Impossible States					•						•	•	
Dot Notation, Naming and Readability										•			
Standard Coding Style						•							
Avoiding Busy Waits in Select Statement												•	
Simple Coding Paradigms				•		•							
Reusable or Marketable Software Modules					•			•					
Hardware Error Detection													•
Customized Run-Time Systems									•			•	
Parameters to Analyze Tasking Performance									•			•	
Analysis of User Requirements								•					

Figure 5-8. Methodology Course Modules Cross-referenced with Future Case Studies

Section 6

INDUSTRIAL TRAINING SURVEY RESULTS

6.1 Survey Development and Distribution

The following sections describe the purpose of the Industrial Training Survey, the development of survey questions, and the survey distribution and administration procedures.

6.1.1 Purpose

The purpose of the Industrial Training Survey was twofold. First, to assess the present level of training activity, both in terms of extent and in-house facilities within the embedded systems community. And second, to document the use and propagation of software development policies and procedures within companies doing large-scale embedded systems development.

6.1.2 Development of Survey Questions

In order for SofTech's Model Ada Training Curriculum, described in Section 5, to meet the objective of the Army's Ada Program, its implementation must be consistent with widely accepted training methods as well as the facilities available for instructional purposes in industry. The Industrial Training Survey was designed to determine the following things:

1. What facilities are presently available in industry for training embedded systems programmers? Do companies normally have their own training staffs, classrooms, audio visual equipment, etc.? (Questions 1-19)

2. How receptive is the embedded systems community to internal and contracted training. Is training a commonplace and accepted practice? (Questions 20-23)

3. What types of training are generally considered effective in the training of embedded systems programmers in languages and design practices? (Questions 23-31)

4) How are the policies and procedures for software development developed, implemented, introduced to staff, audited, and updated? (Questions 32-37)

6.1.3 Survey Distribution and Administration

A meeting was held on Feburary 23, 1982 at CECOM headquarters at Ft. Monmouth, New Jersey for all survey participants (see Appendix A for a list of participating companies). At this meeting the general purpose of the survey was discussed and participants had the opportunity to respond to each item of the survey. Changes in the survey which resulted from this meeting were incorporated and the surveys were distributed on March 1, 1982. The final version of the survey is found in Appendix C. Surveys were completed by personnel directly responsible for training.

6.2 Survey Findings

A total of 20 Training Surveys were distributed with six companies responding (30% returned). Figure 6-1 shows summarized source data indicating percentages of responses for each survey. These percentages are calculated on the number of surveys returned.

Briefly, the Industrial Training Survey reveals that:

a. 66% of the companies surveyed have a full-time training staff, of which 83% have B.S. degrees.

b. 83% offer in-house courses on a regular basis.

c. 100% have classroom and auditorium facilities.

d. 83% have online access to a computer system for instructional purposes.

e. 100% have video cassette recorders and monitors.

f. 100% contract with outside vendors for training.

g. Most companies consider lecture/workshop, formal courses, and video supported classes to be the most effective formats for training programmers in new languages. On-the-job training and self-paced instruction are the least effective formats.

h. The lecture/workshop format is considered to be the most effective means of training embedded systems personnel in program management, system analysis and design, and system architecture. On-the-job training and self-paced instruction are the least effective formats.

i. All respondents indicated that they have documented policies and procedures for software development, and that these policies and procedures were formulated by internal committee or study group.

j. Internally developed courses and the project leaders are considered the two most effective methods of introducing software development policies to the staff.

k. Ada seminars and self-instruction are presently the two most prevalent forms of Ada training.

ADA SOFTWARE DESIGN METHOD FORMULATION
INDUSTRIAL TRAINING SURVEY

ALL QUESTIONS ON THIS SURVEY REFER TO THE DIVISION LEVEL

PART I: EMBEDDED SYSTEMS TRAINING

1. What are the responsibilities of your training department? (Check as many as appropriate.)

83.33%	To procure training from outside sources	66.67%	To develop training materials
83.33%	To coordinate the internal development of training materials	66.67%	To develop and provide training
		----	Other (explain) _____

2. Is your training department composed of:

33.33%	a. Full time staff only
33.33%	b. Part time and full time staff
33.33%	c. Part time only
----	d. Other (explain) _____

3. In terms of training functions does your training department have individuals who (Check as many as appropriate.)

83.33%	a. Identify training needs	50.0%	e. Design and develop training courses and support course materials
50.0%	b. Design training courses	33.33%	f. Perform all functions
67.67%	c. Develop training materials	----	g. Other (explain)_____
67.67%	d. Provide instruction		

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 1 of 10)

IN GENERAL, WHAT ARE THE MINIMUM QUALIFICATIONS OF YOUR COURSE DEVELOPERS?

4. Educational Background:
   83.33% a. BS
   16.67% b. MS
   -0- c. PhD
   -0- d. Other --

5. Technical Experience:
   ____ a. 0-3 yrs.
   66.67% b. 3-5 yrs.
   33.33% c. Over 5
   -0- d. Other --

6. Teaching Experience:
   66.67% a. 0-3 yrs.
   33.33% b. 3-5 yrs.
   -0- c. Over 5
   -0- d. Other --

7. Do you consider any other qualifications when selecting course developers? Yes X No ____
   Explain Expert in the subject, presentation skills, background and qualifications in field, recommendations from others.

IN GENERAL WHAT ARE THE MINIMUM QUALIFICATIONS OF YOUR INSTRUCTORS?

8. Educational Background:
   83.33% a. BS
   16.67% b. MS
   -0- c. PhD
   -0- d. Other --

9. Technical Experience:
   16.67% a. 0-3 yrs.
   50.0% b. 3-5 yrs.
   33.33% c. Over 5
   -0- d. Other --

10. Teaching Experience:
    66.67% a. 0-3 yrs.
    33.33% b. 3-5 yrs.
    -0- c. Over 5
    -0- d. Other --

11. Do you consider any other qualifications when selecting course instructors? Yes ____ No ____
    Explain Expert in the subject of the course, background in computer/military systems, background and qualifications in field, recommendations from others.

SOFTECH

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 2 of

1094-2.1

12. How do you train your educational staff? (Check as many as appropriate.)

66.67% a. Internally through educational department

33.33% b. Internally through technical staff

33.33% c. Contracted training

---- d. Other (explain) _____

13. Does your company offer in-house courses on a regular basis? Yes 83.33% No 16.67%

14. Approximately, how many different courses do you offer per year? ____

6-6

15. In general, are the instructors for these courses: (Check as many as appropriate.)

50.0% a. Full-time training staff

83.33% b. Technical staff

0.0% c. Marketing support technical staff

---- d. Other (explain) _____

16. What is the normal duration of your in-house courses?

33.33% a. 1-2 days

83.33% b. 3-5 days

16.67% c. 6-10 days

16.67% d. over 10 days

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 3 of 10)

17. What is the average class size for an in-house course?

16.67%	a. Less than 10	33.33%	c. 16-30
83.33%	b. 10-15	-0-	d. More than 30

18. When are your in-house courses offered?

33.33%	a. During working hours	66.67%	c. Both
-0-	b. After working hours		

19. What internal facilities are available for company sponsored training? (Check as many as appropriate.)

- 100.0 % a. Classrooms (0-15 people)
- 100.0 % b. Classrooms (16-30 people)
- 66.67% c. Classrooms (over 30 people)
- 100.0 % d. Auditoriums (over 75 people)
- 83.33% e. Laboratories (online access to system)
- 100.0 % f. Video cassette recorders and monitors
- 66.67% g. 16mm or 35mm projection facilities
- 83.33% h. Overhead Projectors
- 33.33% i. Other (please specify) _____

20. Do you contract with outside sources for facilities when necessary? Yes 66.67% No 33.33%

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 4 of 10)

1094-2.1

21. How are your in-house courses evaluated? (Check as many as appropriate.)

100.0 % a. Informal feedback

100.0 % b. Written evaluation from students

16.67% c. Professional review

-0- d. Not evaluated

--- e. Other (explain) _____

22. Does your company ever contract without outside vendors for technical training? Yes 100.0% No _____

6-8

23. What form(s) of training do you purchase? (Check as many as appropriate.)

_____ a. Courses

83.33% 0-3 days

66.67% 3-5 days

33.33% 5-10 days

16.67% Over 10 days

100.0 % b. Seminars/symposia

83.33% c. Videotapes

50.0 % d. Workshops

66.67% e. Tutorial texts

_____ f. Other _____

24. Does your company provide a training curriculum oriented towards specific categories/levels of embedded computer systems personnel? Yes 33.33% No 66.67%

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 5 of 10)

25. Who is responsible for providing in-house training to embedded computer systems personnel? (Check as many as appropriate.)

83.33% a. The department requiring training

66.67% b. An educational organization within your division

-0- c. A corporate educational organization

--- d. Other (explain) _____

26. What percent of the training of embedded computer systems personnel is provided by each of the following:

	0-20%	21-40%	41-60%	61-80%	81-100%
a. In-house courses	16.67	50.0	33.33	0	0
b. College courses	50.0	16.67	16.67	16.67	0
c. On the job training	50.0	33.33	0	16.67	0
d. Contracted with outside vendor	50.0	0	0	0	0
e. Contract/projected oriented training	33.33	16.67	0	0	0
f. Other (explain)	0	0	0	0	0

27. What instructional formats has your company used with embedded systems personnel? (Check as many as appropriate.)

83.33% a. Lecture

50.0 % b. Workshop

100.0 % c. Lecture/workshop

16.67% d. Computer aided instruction

66.67% e. Online exposure

50.0 % f. Self-paced instruction

83.33% g. Videotape

33.33% h. Film

---- i. Other (explain) _____

_____

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 6 of 10)

28. Which formats have you found most effective in the training of programmers in new languages?

a. Lecture/workshop, Formal Courses, Video Supported Classes

b. Online Exposure, Computer Aided Instruction

29. Which formats have you found least effective in the training of programmers in new languages?

a. On-the-job Training, Self-paced instruction

b. Lecture, Video tapes

30. Which formats have you found most effective in the training of embedded systems personnel in program management, system analysis and design, and system architecture?

a. Lecture/Workshop

b. Contracted Instructors

31. Which formats have you found least effective in the training of embedded systems personnel in program management, system analysis and design, and system architecture?

a. On-the-job training, Self-paced instruction

b. Lecture, Internally Developed Videotapes

PART II: SOFTWARE DEVELOPMENT POLICIES

IF YOUR COMPANY HAS DOCUMENTED POLICIES AND PROCEDURES FOR SOFTWARE DEVELOPMENT, ANSWER THE FOLLOWING QUESTIONS:

32. How were these policies and procedures established? (Check as many as appropriate.)

100.0 % a. Internal committee or study group

16.67% b. Internal consultant(s)

16.67% c. Outside consultant(s)

16.67% d. Other _____

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 7 of 10)

33. How were these policies and procedures implemented? (Check as many as appropriate.)

66.67% a. Pilot project

83.33% b. Internally developed courses

-0- c. Contracted training

100.0 % d. Project leader/supervisor

100.0 % e. Printed materials

-0- f. Other _____

34. How are new staff introduced to your software development policies and procedures? (Check as many as appropriate.)

50.0 % a. Internally developed courses

-0- b. Contracted training

100.0 % c. Project leader/supervisor

66.67% d. Printed materials

--- e. Other (explain) _____

35. Rate the following methods of introducing software development policies as to effectiveness.

	Not Effective	Effective	Very Effective
a. Internally developed courses	16.67%	50.0 %	33.33%
b. Contracted training	16.67%	16.67%	-0-
c. Project leader/supervisor	-0-	66.67%	33.33%
d. Printed materials	33.33%	50.0 %	-0-

e. Other (explain) _____

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 8 of 10)

1094-2.1

36. How do you audit your software development policies and procedures? (Check as many as appropriate.)

83.33% a. Internal Quality Assurance

100.0 % b. Project leader/supervisor evaluation

100.0 % c. Design reviews

66.67% d. Walkthroughs

16.67% e. Not audited

--- f. Other (explain) _____

37. If policies and procedures are updated, how are these changes communicated to the staff? (Check asmany as appropriate.)

66.67% a. Internal developed courses

-0- b. Contracted training

100.0 % c. Project leader/supervisor

100.0 % d. Printed materials

-0- e. Other (explain) _____

6-12

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 9 of 10)

1094-2.1

PART III: Ada TRAINING

IF ANY OF YOUR STAFF HAVE PARTICIPATED IN Ada TRAINING, ANSWER THE FOLLOWING QUESTIONS:

38. Approximately, what percentage of your staff has participated in Ada training? varying 2-15%

39. What form(s) of Ada training has your staff received? (Check as many as appropriate.)

- 83.33% a. Self
- 100.0 % b. Ada seminars
- 66.67% c. University sponsored course
- 16.67% d. Government sponsored course
- 33.33% e. Videotapes
- -0- f. Film
- g. In-house course
  - 50.0 % 1-3 days
  - 33.33% 3-5 days
  - -0- More than 5 days
- ---- h. Other (explain) _____

6-13

40. In contrast to other courses, do you believe Ada courses will require:

- -0- a. Less time to develop
- 16.67% b. Approximately the same time to develop
- 83.33% c. More time to develop
- -0- d. Less actual class time
- -0- e. Approximately the same class time
- 83.33% f. More actual class time

41. Thank you for completing this Survey. If there are additional comments you wish to make, please feel free to make them below.

SOFTECH

Figure 6-1. Summarized Source Data Industrial Training Survey (Page 10 of 10)

END

DATE
FILMED

3-83

DTIC